SYSTÈMES D'EXPLOITATION

Notes de cours

Pr. Omar Megzari

megzari@fsr.ac.ma

Faculté des Sciences de Rabat,

Département d'Informatique

1 Introduction

Ce cours présente les principaux points théoriques du fonctionnement des systèmes d'exploitation. Il les illustre par un certain nombre d'exemples de mise en œuvre qu'il tire essentiellement du système Unix et accessoirement de Windows.

L'histoire de l'informatique est très brève – les ordinateurs sont nés avec la seconde guerre mondiale – et pourtant, elle a connu des grandes évolutions. À leur apparition, les ordinateurs étaient très coûteux et réservés aux grandes entreprises; celles-ci n'en possédaient au départ que quelques exemplaires. Ces ordinateurs « centraux » sont rapidement devenu un auxiliaire d'administration et ils se sont diffusés dans les différents services (*departments* en anglais) : financier, comptabilité, etc. Pour rendre l'informatique plus adaptée et plus abordable, des fabricants se sont alors mis à produire des mini-ordinateurs « départementaux ». Ces ordinateurs fonctionnaient avec des systèmes d'exploitation qui leur étaient propres, à chaque machine ou à chaque constructeur, par exemple, MVS pour IBM ou VMS pour DEC.

Aujourd'hui, l'informatique, aussi bien dans les entreprises, que dans la recherche ou l'enseignement, utilise des machines plus petites, fonctionnant avec des systèmes d'exploitation à caractère universel. Parmi ces systèmes d'exploitation, deux se distinguent particulièrement, un système mono-utilisateur, Windows, et un autre multi-utilisateurs et multitâches, Unix. D'une manière grossière – et contestable avec l'apparition des réseaux – on peut affirmer que le premier système est destiné à des ordinateurs individuels, tandis que l'autre est réservé au travail en groupe. Les systèmes actuels gèrent, par ailleurs une interface graphique, avec comme pionnier le Finder du Macintosh. Les systèmes d'exploitation actuels ont intégré de façon généralisée le multitâches et le service à plusieurs utilisateurs avec la généralisation des architectures client-serveur, par exemple avec OS/2 d'IBM et Windows/NT.

Parmi ces systèmes, Unix, qui est le plus ancien, est celui qui offre le plus de richesses, le plus d'homogénéité et le plus de souplesse; il dispose, dans les versions standards, d'extensions pour les réseaux et pour le graphique. Pour cette raison, nous l'avons choisi comme le centre de ce cours. Par ailleurs, le système MS-DOS puis Windows, en évoluant, ont incorporé beaucoup de caractéristiques de leur prédécesseur. Les noyaux de ces systèmes se modifieront certainement avec l'évolution des techniques. Cependant, les principes sur lesquels ils se fondent, et à plus forte raison, leur « décor », devraient rester relativement stables, au moins pour les quelques années à venir.

L'étude des systèmes d'exploitation forme une part très importante de l'informatique comme discipline et, à la différence des ses autres domaines, c'est une part qui lui est propre. Ceci au contraire de l'algorithmique ou de la logique, par exemple, qui se partagent avec les mathématiques. C'est aussi une discipline technique qui plus encore que les autres est sujette au renouvellement.

On peut diviser les systèmes d'exploitation classiques en quatre parties principales :

1. Les **processus**, qui correspondent à l'exécution des programmes. Ces processus pouvant s'exécuter simultanément dans un système multitâche. Le système a pour fonction de les créer, de les gérer, de les synchroniser, ainsi que de leur permettre de communiquer entre eux;
2. La **gestion de la mémoire**, qui permet de transférer les programmes et les données nécessaires à la création des processus, d'un support secondaire, par exemple un disque, vers un support central, où a lieu l'exécution des processus. Le système devra garder la trace des parties utilisées et libres de la mémoire ainsi que gérer les transferts entre les mémoires principale et secondaire;
3. Le **système de fichiers**, qui offre à l'utilisateur une vision homogène et structurée des données et des ressources : disques, mémoires, périphériques. Le système gère la création des fichiers, leur destruction, leur correspondance avec les dispositifs physiques, ainsi qu'un certain nombre d'autres caractéristiques, telles que la protection. Il les organise enfin, en général, en une structure arborescente;
4. **Les entrées-sorties**, qui correspondent aux mécanismes qu'utilisent les processus pour communiquer avec l'extérieur. Ces entrées-sorties font largement appel aux couches les plus proches du matériel, et dont le système tente de masquer les particularités aux utilisateurs.

Les systèmes d'exploitation modernes intègrent par ailleurs d'autres caractéristiques. Ces dernières concernent notamment deux évolutions majeures des systèmes informatiques. La première est l'interconnexion des différentes machines et des différents systèmes par des réseaux locaux ou étendus. La seconde est la disparition des écrans de textes et leur remplacement par des dispositifs à fenêtres multiples disposant de propriétés graphiques. Ces deux techniques sont, de plus, étroitement imbriquées. Les systèmes d'exploitation fonctionnent donc, ou vont fonctionner, en réseau et ils consacreront une part importante de leurs tâches à gérer le fenêtrage, le graphisme, et les interactions entre les différentes machines. Ce cours complète les 4 parties précédentes par des études sur :

5. Les **réseaux** d'ordinateurs, avec les protocoles de communication, d'interconnexion et d'application. Les réseaux permettent de mettre en œuvre une nouvelle architecture informatique fondée sur des clients et des serveurs;
6. Les **systèmes répartis** avec les protocoles d'appels de procédures à distance qui leur sont associés. Les systèmes répartis actuels trouvent des applications à des architectures clients-serveurs de fichiers ou d'applications, tels que des bases de données.
7. Les **systèmes de fenêtrage** graphique, ainsi que le modèle de serveur d'écran.

Le système d'exploitation correspond à l'interface entre les applications et le matériel. Le programmeur d'applications n'aborde que rarement – sinon jamais – son code interne. Il l'utilise par l'intermédiaire d'« appels système ». Les appels systèmes sont souvent accessibles à partir d'un langage de programmation, notamment en C avec le système Unix. Ces appels permettent d'effectuer la plupart des opérations sur les entités du système d'exploitation et, par exemple, de créer et détruire des processus, des fichiers, de réaliser des entrées-sorties, etc. Une terminologie tend à s'imposer pour dénommer l'ensemble des appels système, qu'ils concernent un système d'exploitation ou n'importe quelle d'application informatique : les API (*Application Programming Interface*).

Un utilisateur peut lui aussi – dans une certaine mesure – manipuler un système d'exploitation, sans pour autant avoir à créer un programme. Il le fait par l'intermédiaire d'un interprète de commandes (un « shell » en anglais) muni d'une syntaxe et éventuellement programmable. Cet interprète peut accepter les lignes de commandes comme sous MS-DOS ou sous Unix. Il peut aussi gérer les « métaphores » graphiques comme avec les Macintoshes, Windows ou X-Window.

2 Bibliographie

La bibliographie sur les systèmes d'exploitation est très abondante et elle se renouvelle très rapidement. Elle comprend à la fois des revues de recherche, des ouvrages pédagogiques et des ouvrages sur la programmation et l'utilisation d'un système particulier. La liste que nous donnons n'est absolument pas exhaustive. Par ailleurs, cette recherche est largement passée des laboratoires universitaires à ceux de quelques industriels : Microsoft et IBM notamment. Notre liste fournit seulement les références que nous pensons être les plus utiles.

Littérature générale sur les systèmes d'exploitation

A. Tanenbaum, *Modern Operating Systems,* Prentice–Hall, 1992, est une référence générale très pédagogique. *Distributed Computer Systems*, Prentice Hall, 1994, examine plus en détail les systèmes répartis. *Operating Systems, 2nd ed.,* Prentice-Hall, 1997, est une référence par le même auteur qui comprend le code, très formateur, d'un système d'exploitation voisin d'Unix. Cette édition ne comprend pas les systèmes répartis.

A. Silberschatz and P. Galvin, *Operating System Concepts*, 5th ed., Addison Wesley, 1997, est un ouvrage plus conceptuel et plus théorique que le précédent. Il reste néanmoins très clair. Il est traduit en français chez Addison-Welsey sous le titre *Principes des systèmes d'exploitation* mais peut être pas l'édition la plus récente.

Sur les noyaux de systèmes d'exploitation commerciaux

M. Bach, *La conception du système Unix*, Masson, a longtemps été l'ouvrage de référence. Sa rédaction est extrêmement lourde – à éviter après le dessert – et elle est doublée d'une traduction maladroite. Cet ouvrage est cependant une mine de renseignements pour ceux qui veulent connaître les algorithmes internes d'Unix en détail.

L'ouvrage qui précède a inspiré les firmes conceptrices d'autres systèmes. Ceci a donné lieu à plusieurs ouvrages, en général plus clairs et mieux écrits que leur ancêtre :

- H.M Deitel et M.S. Kogan, *La conception d'OS/2*, Addison-Wesley, 1992.
- Helen Custer, *Au cœur de Windows NT*, Microsoft Press, 1993.

Sur la programmation des systèmes d'exploitation

Le `man` d'Unix est la meilleure référence pour le programmeur. Il n'y a rien d'équivalent sur papier.

J.M. Rifflet, *La programmation sous Unix*, 3e éd., McGraw-Hill, 1993, est une bonne référence et un ouvrage assez complet.

Charles Petzold, *Programming Windows 95*, Microsoft Press, 1996, est une bonne référence pour apprendre la programmation Windows. Elle a été la première du genre. Actuellement, il y a des dizaines d'ouvrages équivalents.

Apple Computer, *Inside Macintosh Series*, Addison-Wesley, 1992, 1993, 1994, est une série traitant des caractéristiques du Macintosh. Elle détaille aussi bien les mécanismes internes que les méthodes de programmation.

Sur l'utilisation du système Unix

R.S. Bourne, *Le système Unix*, InterEditions, est une référence antique, mais qui reste un modèle de clarté. L'auteur est le concepteur du premier interprète de commande d'Unix. La traduction comprend beaucoup de fautes, par exemple dans les listings de programmes.

B. Kernighan et R. Pike, *L'environnement Unix*, InterEditions, s'axe plutôt sur les outils d'Unix. Il est plus difficile à lire que le précédent et il privilégie parfois la « bidouille » info-maniaque.

J.L. Nebut, *Unix pour l'utilisateur : Commandes et langages de commandes*, Technip, 1990, est une bonne référence sur les outils d'Unix.

3 L'Ordinateur

De nos jours, l'ordinateur est devenu un outil d'usage courant. Mais comment fonctionne cette machine si intrigante? Sans instruction pour le faire fonctionner, l'ordinateur est une grosse boîte de métal et de plastique qui ne sert à rien. Tout comme n'importe quel appareil ménager, l'ordinateur a besoin d'instructions. Elles lui sont transmises par le clavier ou la souris. Tout comme vous conduisez une voiture sans en maîtriser chaque élément mécanique, il n'est pas nécessaire de tout connaître sur le fonctionnement d'une voiture pour être capable de l'utiliser.

L'ordinateur est une machine qui traite des informations (des textes, des photos, des images) par l'intermédiaire de ses circuits électroniques et de ses programmes informatiques.

Pour transférer ces informations à l'ordinateur, nous utilisons des accessoires tels que le clavier, la souris, le numériseur et la caméra numérique. Nous les appelons des périphériques d'entrée.

Pour bien percevoir les informations transférées et être capables d'en apprécier les modifications, nous utilisons des périphériques de sortie tels que l'écran et l'imprimante.

Pour que l'ordinateur puisse intégrer les informations qui lui sont envoyées, il faut qu'il puisse les comprendre. Pour ce, il faudra qu'elles soient modifiées en langage machine. En fait, si nous appuyons sur une touche du clavier, l'ordinateur recevra une série d'impulsions électriques. Ces impulsions seront interprétées par l'ordinateur sous forme de séries de (0 et 1) qui compose le langage informatique. Ces impulsions, ou chiffres, sont appelées des «bits».

En appuyant sur une touche, le clavier envoie sous forme d'impulsions électriques, une série de 0 et de 1, toujours en nombre exact de huit, que l'on appelle des octets. La lettre B, par exemple, est convertie en une série de huit chiffres : 01000010.

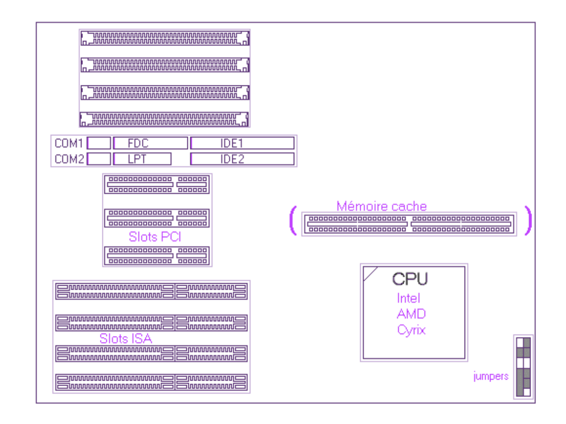
L'origine du mot «ordinateur» vient de la firme IBM. Celle-ci demanda en 1954 à un professeur de lettres à Paris de trouver un mot pour désigner ce que l'on appelait vulgairement un «calculateur» (traduction littérale de *computer* en anglais). Toute machine capable de manipuler des informations binaires (0 et 1) peut être qualifiée d'ordinateur.

Si nous excluons les périphériques d'entrée et de sortie, que nous avons brièvement énumérés auparavant, un ordinateur est composé :

- d'une unité centrale (le boîtier);
- de périphériques **internes** (cartes de son, carte vidéo, ...);
- d'un lecteur de disquettes, d'un lecteur de CD-ROM ou de DVD-ROM;
- éventuellement, de cartes d'extension diverses, ...

3.1 La carte mère

Schéma d'une carte mère :



La carte mère est le principal constituant de l'ordinateur. C'est sur cette carte que sont connectés les autres éléments :

- Le microprocesseur (cerveau de l'ordinateur);
- La mémoire (RAM : *Random Access Memory*, la mémoire cache);
- Le disque dur, le lecteur de CD-ROM, le lecteur de disquettes;
- Les périphériques internes : carte de son, carte vidéo.

3.2 Le microprocesseur

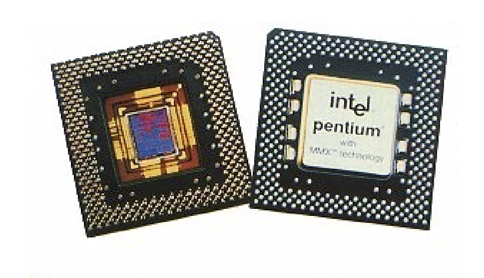
Le premier microprocesseur (Intel 4004) a été inventé en 1971. Depuis, la puissance des microprocesseurs augmente exponentiellement. Quels sont donc ces petits morceaux de silicium qui dirigent nos ordinateurs?

Le processeur (CPU) est le cerveau de l'ordinateur, c'est lui qui coordonne le reste des éléments, il se charge des calculs, bref, il exécute les instructions qui ont été programmées.

Toutes ces opérations sont des informations numériques. Les microprocesseurs utilisent des petits transistors pour faire des opérations de base; il y en a plusieurs millions sur un seul processeur.

Les principaux éléments d'un microprocesseur sont :

- une horloge qui rythme le processeur. À chaque TOP d'horloge, le processeur effectue une instruction. Ainsi plus l'horloge a une fréquence élevée, plus le processeur effectue d'instructions par seconde (MIPS : Millions d'instruction par seconde). Par exemple un ordinateur ayant une fréquence de 100 mégahertz (MHz) effectue 100 000 000 d'instructions par seconde;
- une unité de gestion des bus qui gère les flux d'informations entrant et sortant;
- une unité d'instruction qui lit les données, les décode puis les envoie à l'unité d'exécution;
- une unité d'exécution accomplit les tâches données par l'unité d'instruction.



Le processeur travaille, en fait, grâce à un nombre très limité de fonctions comme des expressions logiques (ET, OU, NON, etc.), des expressions mathématiques (addition, soustraction, multiplication, etc.). Celles-ci sont directement câblées sur les circuits électroniques. Il est impossible de mettre toutes les instructions sur un processeur car celui-ci est limité par la taille de la gravure. Ainsi pour mettre plus d'instructions il faudrait un processeur ayant une très grande surface. Or le processeur est constitué de silicium et celui-ci coûte cher, et d'autre part il chauffe beaucoup. Le processeur traite donc les informations compliquées à l'aide d'instructions simples.

3.3 La mémoire cache

La mémoire cache permet au processeur de se «rappeler» les opérations déjà effectuées auparavant. Elle est utilisée par le microprocesseur pour conserver temporairement des instructions élémentaires. En effet, elle stocke les opérations effectuées par le processeur, afin que celui-ci ne perde pas de temps à recalculer des calculs déjà faits précédemment. La taille de la mémoire cache est généralement de l'ordre de 512 kilo-octets (Ko).

3.4 La mémoire vive

La mémoire vive, généralement appelée RAM (*Random Access Memory*, traduisez *mémoire à accès aléatoire*) ce qui signifie que l'on peut accéder instantanément à n'importe quelle partie de la mémoire, permet de stocker des informations pendant tout le temps de fonctionnement de l'ordinateur. Par contre, cette mémoire est détruite lors de la mise hors-tension de l'ordinateur, contrairement à une mémoire de masse comme le disque dur qui garde les informations même lorsqu'il est hors tension. La mémoire vive contient les données et les instructions des applications en cours.

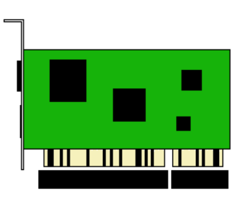
3.5 La mémoire morte (ROM)

Mémoire permanente contenant des microprogrammes enregistrés sur des puces électroniques de la carte mère (ou *mother board*) contenant les routines de démarrage du micro-ordinateur. ROM (*Read Only Memory*, dont la traduction est *mémoire en lecture seule*) est appelée aussi parfois *mémoire non volatile*, car elle ne s'efface pas lors de la mise hors tension du système. En effet, ces informations ne peuvent être stockées sur le disque dur étant donné que les paramètres du disque (essentiels à son initialisation) font partie de ces données vitales à l'amorçage.

3.6 Les fentes d'extension

Les fentes ou «slots» d'extension sont des réceptacles dans lesquels on peut enficher des cartes. Il en existe de trois types : les cartes ISA (les plus lentes fonctionnant en 16 bits), les

cartes PCI (beaucoup plus rapides fonctionnant en 32 bits), et les cartes AGP (les plus rapides). Ils se branchent, grâce à des nappes, sur les broches prévues à cet effet sur la carte mère.



3.7 Le disque dur

Le disque dur est l'organe du PC servant à conserver les données de manière permanente, contrairement à la RAM, qui s'efface à chaque redémarrage de l'ordinateur. Il a été inventé au début des années 50 par IBM.

3.7.1 Le fonctionnement interne

Un disque dur est constitué non pas d'un seul disque, mais de plusieurs disques rigides (en anglais *hard disk* signifie *disque dur*) en métal, en verre ou en céramiques empilés les uns après les autres à une très faible distance les uns des autres. Ils tournent très rapidement autour d'un axe (à plusieurs milliers de tours par minute actuellement) dans le sens inverse des aiguilles d'une montre. Un ordinateur fonctionne de manière binaire. Il faut donc stocker les données sous forme de 0 et de 1. C'est pourquoi les disques sont recouverts d'une très fine couche magnétique de quelques microns d'épaisseur, elle-même recouverte d'un film protecteur.

La lecture et l'écriture se font grâce à des têtes (*head*) situées de part et d'autre de chacun des plateaux (un des disques composant le disque dur). Ces têtes sont des électroaimants qui se baissent et se soulèvent (elles ne sont qu'à quelques microns de la surface, séparées par une couche d'air provoquée par la rotation des disques qui crée un vent d'environ 250 km/h) pour pouvoir lire l'information ou l'écrire. De plus ces têtes peuvent balayer latéralement la surface du disque pour pouvoir accéder à toute la surface...



Cependant, les têtes sont liées entre-elles et seulement une seule tête peut lire ou écrire à un moment donné. On parle donc de cylindre pour désigner l'ensemble des données stockées verticalement sur la totalité des disques.

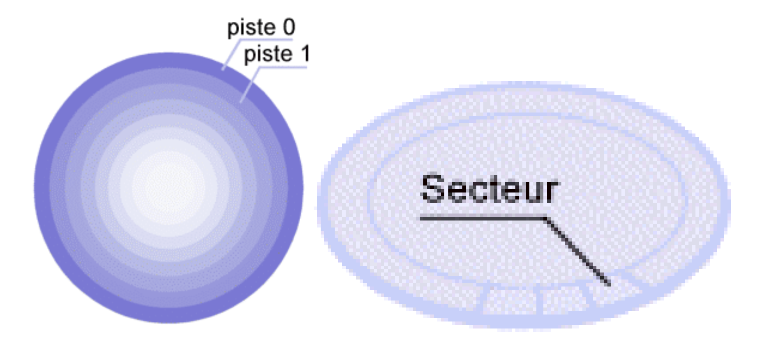
L'ensemble de cette mécanique de précision est contenu dans un boîtier totalement hermétique, car la moindre particule peut détériorer l'état de surface du disque. Vous pouvez donc voir sur un disque des opercules permettant l'étanchéité, et la mention "*Warranty void if removed*" qui signifie littéralement "*la garantie expire si retiré*", seul les fabricants peuvent en vérifier le contenu (à l'intérieur de salles blanches : exemptes de particules).

3.7.2 La lecture et l'écriture

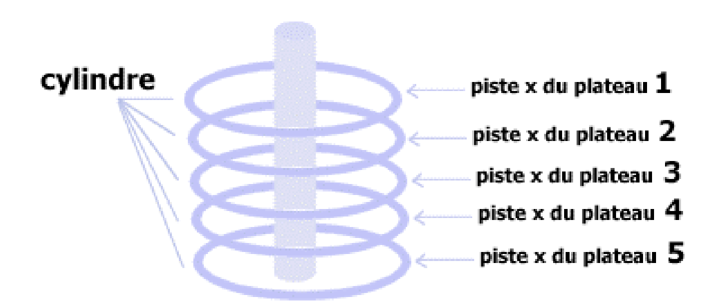
Les têtes de lecture/écriture sont dites "inductives", c'est-à-dire qu'elles sont capables de générer un champ magnétique. C'est notamment le cas lors de l'écriture, les têtes en créant des champs positifs ou négatifs viennent polariser la surface du disque en une très petite zone, ce qui se traduira lors du passage en lecture par des changements de polarité induisant un courant dans la tête qui sera ensuite transformés par un convertisseur analogique numérique (CAN) en 0 et en 1 compréhensibles par l'ordinateur.

Les têtes commencent à inscrire des données à la périphérie du disque (piste 0), puis avancent vers le centre. Les données sont organisées en cercles concentriques appelés "pistes", créées par le formatage de bas niveau.

Les pistes sont séparées en quartiers (entre deux rayons) que l'on appelle *secteurs*, c'est la zone dans laquelle on peut stocker les données (512 octets en général).



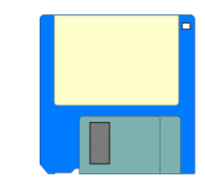
On appelle cylindre l'ensemble des données situées sur une même pistes de plateaux différents (c'est-à-dire à la verticale les unes des autres) car cela forme dans l'espace un "cylindre" de données.

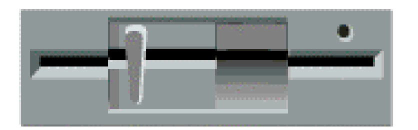


On appelle *cluster* la zone minimale que peut occuper un fichier sur le disque. En effet le système d'exploitation exploite des ***blocs*** qui sont en fait plusieurs ***secteurs*** (entre 1 et 16 secteurs). Un fichier minuscule devra donc occuper plusieurs secteurs (un cluster).

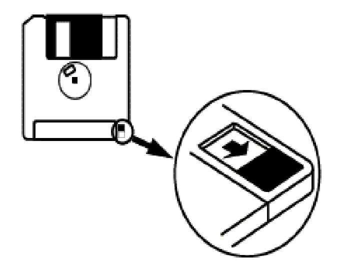
3.8 La disquette

Constituée d'une rondelle de plastique souple recouverte d'un carré de plastique dur pour la protéger, la disquette à pour avantage d'être amovible, c'est-à-dire que l'on peut l'insérer et l'enlever du lecteur très facilement. Elle existe principalement en deux formats: 720 Ko et 1,44 Mo.





Pour protéger la disquette, c'est-à-dire être certain de ne pas en effacer le contenu, il faut rabattre, de cette façon, la petite pièce de plastique.



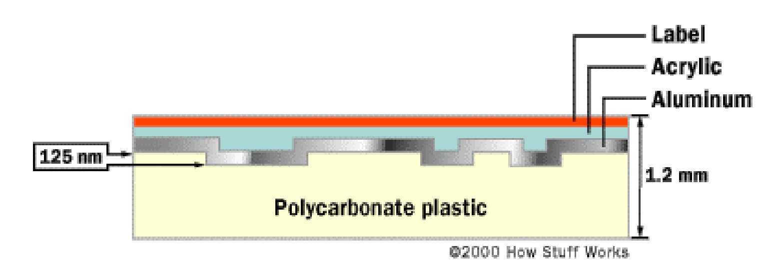
3.9 Le CD-ROM

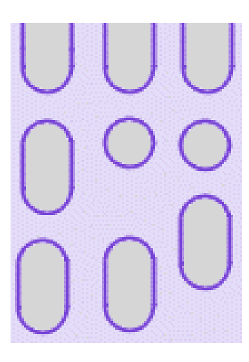
Il utilise des disques portatifs de grande capacité au format pratique, de plus en plus utilisé pour la vente de logiciels. Il remplacera le lecteur de disquettes dans les prochaines années.

Le CD-ROM (Compact Disc - Read Only Memory) est un disque optique de 12 cm de diamètre et de 1mm d'épaisseur, permettant de stocker des informations numériques, c'est-à-dire correspondant à 650 Mo de données informatiques (correspondant à 300000 pages dactylographiées) ou bien jusqu'à 78 min de données audio. Le Compact Disc a été inventé par Sony © et Philips ©.

3.9.1 La composition d'un CD-ROM

Le CD est constitué de matière plastique, recouvert d'une fine pellicule métallique d'aluminium sur une des faces. Les pistes sont gravées en spirales, ce sont en fait des alvéoles d'une profondeur de 125nm et espacées de 1,6µ. Ces alvéoles forment un code binaire, une alvéole correspond à un 0, un espace à un 1.



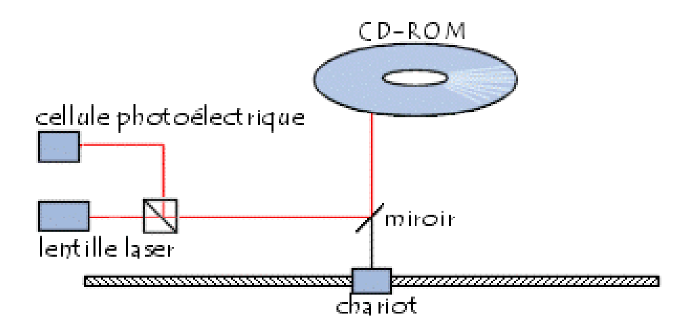


Exemple : prenons la séquence suivante : 110010101. Celle-ci correspond sur le CD-ROM à deux espaces, deux trous, un espace, un trou, un espace, un trou, un espace.

On a ainsi une séquence binaire que le lecteur parcourt grâce à un laser ; celui-ci est réfléchi lorsqu'il rencontre un espace, il ne l'est pas lorsqu'il rencontre une alvéole.

3.9.2 Le lecteur de CD-ROM

C'est une cellule photoélectrique qui permet de capter le rayon réfléchi, grâce à un miroir semi-réflechissant comme expliqué sur le dessin suivant :



Un chariot permet de déplacer le miroir de façon à pouvoir accéder au CD-ROM en entier.
Il est ainsi possible de stocker sur ce support des musiques, des images, des vidéos, du texte et tout ce qui peut être enregistré de façon numérique.

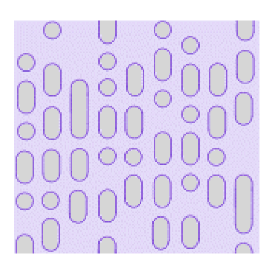
3.9.3 Ses caractéristiques

Le lecteur CD-ROM est caractérisé :

- Par sa vitesse : celle-ci est calculée par rapport à la vitesse d'un lecteur de CD-Audio (150 Ko/s). Un lecteur pouvant atteindre la vitesse de 3000 Ko/s sera caractérisé de 20X (20 fois plus vite qu'un lecteur 1X)
- Par son temps d'accès, on défini le temps moyen que le lecteur prend pour atteindre une partie du CD à une autre.

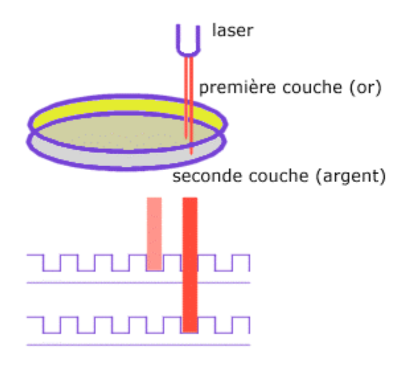
3.10 Le DVD-ROM

Le DVD-ROM (Digital Versatile Disc - Read Only Memory) est une variante du CD-ROM dont la capacité est largement plus grande. En effet, les alvéoles du DVD sont beaucoup plus petite (0,4µ et un espacement de 0.74µ), impliquant un laser avec une longueur d'onde beaucoup plus faible.



Les DVD existent en version "double couche", ces disques sont constitués d'une couche transparente à base d'or et d'une couche réflexive à base d'argent. Dans le but de lire ces deux couches, le lecteur dispose d'un laser à deux intensités :

- une intensité faible ; le rayon se réfléchit sur la surface dorée
- une plus grande intensité permet au rayon de traverser la première couche et de se réfléchir sur la surface argentée.



Il existe 4 types de DVD différents :

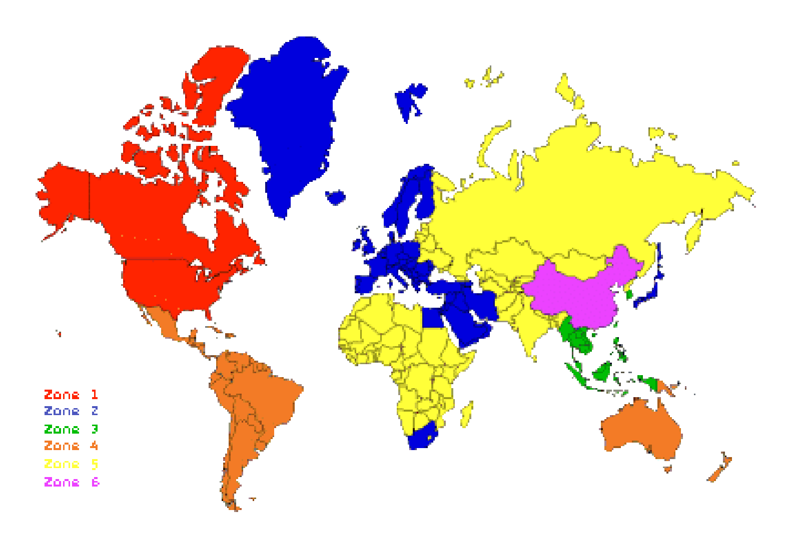
3.10.1.1.1.1.1.1 Type de support	3.10.1.1.1.1.1.2 Capacité	3.10.1.1.1.1.1.3 Temps musical équivalent	3.10.1.1.1.1.1.4 Nombre de CD équivalent
CD	650Mo	1h18 min	1
DVD simple face simple couche	4.7Go	9 h 30	7
DVD simple face double couche	8.5Go	17 h 30	13
DVD double face simple couche	9.4Go	19 h	14

DVD double face double couche	17Go	35 h	26

L'intérêt du DVD touche en priorité le stockage vidéo qui demande beaucoup plus d'espace. Un DVD de 4,7 Go permet de stocker plus de deux heures de vidéo compressées en MPEG-2 (Motion Picture Experts Group), un format qui permet de compresser les images tout en gardant une très grande qualité d'image.

3.10.2 Les zones

Les DVD Vidéo sont conçus pour n'être consultables que dans certaines régions du monde : c'est le découpage en zone (qui "empêche" le piratage). Il est ainsi théoriquement impossible de lire un DVD d'une zone en étant dans une autre. Heureusement, les lecteurs de DVD pour PC peuvent les lire grâce à des utilitaires.



Les premiers graveurs de DVD sont apparus il y a peu de temps. Le seul frein est l'existence de deux normes concurrentes et incompatibles :

- DVD-RAM de Toshiba © et Matsushita © stockant 2.6 Go
- DVD-RW de Sony ©, Philips © et HP © stockant 3 Go

Les deux normes permettent de réinscrire des données jusqu'à 1000 fois.

3.11 Le modem

Le morse a été le premier codage à permettre une communication longue distance. C'est *Samuel F.B.Morse* qui l'a mis au point en 1844. Ce code est composé de points et de tirets (un langage binaire en quelque sorte). Il permettait d'effectuer des communications beaucoup plus rapides que le Pony Express. L'interpréteur était l'homme à l'époque, il fallait toutefois une bonne connaissance du code.

De nombreux codes furent inventés dont le code d'Émile Baudot (portant d'ailleurs le nom de code *Baudot*, les anglais l'appelaient *Murray Code*).

Le 10 mars 1876, le Dr Graham Bell met au point le téléphone, une invention révolutionnaire qui permet de faire circuler de l'information vocale dans des lignes métalliques.

Ces lignes permirent l'essor des télescripteurs, des machines permettant de coder et décoder des caractères grâce au code Baudot (Les caractères étaient alors codés sur 5 bits, il y avait donc 32 caractères uniquement...).

Dans les années 60, le code ASCII (American Standard Code for Information Interchange) est adopté comme standard. Il permet le codage de caractères sur 8 bits, soit 256 caractères possibles.

Grâce aux techniques de numérisation et de modulation autour de 1962 ainsi que de l'essor des ordinateurs et des communications, le transfert de données via modem vit le jour.

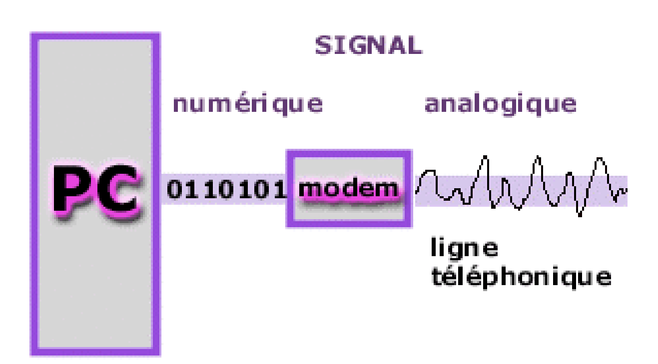
Le modem est le périphérique utilisé pour transférer des informations entre plusieurs ordinateurs (2 à la base) via les lignes téléphoniques. Les ordinateurs fonctionnent de façon digitale, ils utilisent le langage binaire (une série de zéros et de uns), mais les modems sont analogiques. Les signaux digitaux passent d'une valeur à une autre, il n'y a pas de milieu, de moitié, c'est du Tout Ou Rien (un ou zéro). L'analogique par contre n'évolue pas "par pas", il couvre toutes les valeurs. ainsi vous pouvez avoir 0, 0.1, 0.2, 0.3 ...1.0 et toutes les valeurs intermédiaires.

Un piano par exemple marche plus ou moins de façon analogue car il n'y a pas "de pas" entre les notes. Un violon par contre peut moduler ses notes pour passer par toutes les fréquences possibles.

Un ordinateur fonctionne comme un piano, un modem comme un violon. Le modem convertit en analogique l'information binaire provenant de l'ordinateur. Il envoie ensuite ce nouveau code dans la ligne téléphonique. On peut entendre des bruits bizarres si on le volume du son provenant du modem.

Ainsi, le modem module les informations numériques en ondes analogiques; en sens inverse il démodule les données numériques.

C'est pourquoi modem est l'acronyme de MOdulateur/DEModulateur.



3.12 La carte réseau

La carte réseau est utilisée à d'interface physique entre l'ordinateur et le câble. Elle traite les données émises par l'ordinateur, elle les transfère et contrôle le flux de données entre l'ordinateur et le câble. Elle traduit aussi les données venant du câble en octets de façon que l'Unité Centrale de l'ordinateur puisse les comprenne. Enfin, la carte réseau s'insère dans un connecteur d'extensions (slot).

3.12.1 La préparation des données

Les données se déplacent dans l'ordinateur en empruntant des chemins appelés « Bus ». Plusieurs chemins côte à côte font que les données se déplacent en parallèle et non en série (les unes à la suite des autres).

- Les premiers bus fonctionnaient en 8 bits (8 bits de données transportés à la fois)
- L'ordinateur PC/AT d'IBM introduit les premiers bus 16 bits
- Aujourd'hui, la majorité des bus fonctionnent en 32 bits

Toutefois sur un câble les données circulent en série (un seul flux de bits), en se déplaçant dans un seul sens. L'ordinateur peut envoyer **OU** recevoir des informations mais il ne peut pas effectuer les deux simultanément. Ainsi, la carte réseau restructure un groupe de données arrivant en parallèle en données circulant en série (1 bit).

Pour cela, les signaux numériques sont transformés en signaux électriques ou optiques susceptibles de voyager sur les câbles du réseau. Le dispositif chargé de cette traduction est le **Transceiver**.

3.12.2 Le rôle d'identificateur

- La carte traduit les données et indique son adresse au reste du réseau afin de pouvoir être distinguée des autres cartes du réseau.
- Adresses : définies par l'IEEE (Institute of Electrical and Electronics Engineer) qui attribue des plages d'adresses à chaque fabriquant de cartes réseau.
- Elles sont inscrites sur les puces des cartes : procédure appelée « Gravure de l'adresse sur la carte ». Par conséquent, chaque carte a une adresse UNIQUE sur le réseau.

3.12.3 Les autres fonctions de la carte réseau

L'ordinateur et la carte doivent communiquer afin que les données puissent passer de l'un vers l'autre. L'ordinateur affecte ainsi une partie de sa mémoire aux cartes munies d'un Accès Direct à la Mémoire (DMA : Direct Access Memory).

La carte indique qu'un autre ordinateur demande des données à l'ordinateur qui la contient. Le bus de l'ordinateur transfère les données depuis la mémoire de l'ordinateur vers la carte réseau.

Si les données circulent plus vite que la carte ne peut les traiter, elles sont placées dans la mémoire tampon affectée à la carte (RAM) dans laquelle elles sont stockées temporairement pendant l'émission et la réception des données.

3.12.4 Envoi et contrôle des données

Avant que la carte émettrice envoie les données, elle dialogue électroniquement avec la carte réceptrice pour s'accorder sur les points suivants :

- Taille maximale des groupes de données à envoyer
- Volume de données à envoyer avant confirmation
- Intervalles de temps entre les transmissions partielles de données
- Délai d'attente avant envoi de la confirmation
- Quantité que chaque carte peut contenir avant débordement
- Vitesse de transmission des données

Si une carte plus récente, donc plus perfectionnée, communique avec une carte plus lente, elles doivent trouver une vitesse de transmission commune. Certaines cartes ont des circuits leur permettant de s'adapter au débit d'une carte plus lente.

Il y a donc acceptation et ajustement des paramètres propres à chacune des deux cartes avant émission et réception des données.

3.13 Périphériques d'entrée

3.13.1 Le clavier

Le clavier est le plus important périphérique d'entrée de données. Grâce à lui, il est possible de transférer des textes ou encore de donner des ordres à la machine d'effectuer des opérations particulières. De la même façon que sur une machine à écrire, le clavier permet de saisir des caractères (lettres, chiffres, symboles, …).

3.13.2 La souris

La souris permet aussi de saisir une information concernant la position du curseur sur l'écran et par conséquent le choix de l'utilisateur.

3.13.3 Le numériseur

Périphérique d'entrée qui permet, par balayage optique, la restitution d'une image à l'écran de l'ordinateur. Le numériseur est semblable, dans sa forme, au photocopieur, avec toutefois une importante distinction; l'image récupérée par l'appareil est transmise à l'ordinateur au lieu d'être immédiatement imprimée sur du papier.



Ce périphérique permet de numériser du texte et des images, en noir et blanc ou en couleur. La qualité dépend de la résolution. On définit la résolution par le nombre de points par pouce (ppp ou dpi), une image étant composée d'une grande quantité de points. Plus le nombre de points est grand dans un pouce carré, plus la qualité de l'image est bonne.

3.13.4 La caméra numérique

Les caméras numériques sont des appareils photographiques qui ne contiennent pas de film. Les photos sont enregistrées sur une petite disquette au lieu de s'imprégner sur une pellicule. La photographie obtenue pourra être visionnée à partir de l'écran d'un ordinateur, ou encore d'un téléviseur. Le grand avantage de ces nouveaux appareils est leur capacité à transmettre une photo à un ordinateur, par l'intermédiaire d'un fil, pour ensuite l'intégrée à un document.



D'autres avantages de la caméra numérique : il n'est pas nécessaire d'acheter des pellicules et de payer pour en faire le développement. De plus, la photo transmise à l'ordinateur, peut être retouchée. Une imprimante couleur de bonne qualité procurer, par la suite, de bonnes photos sur papier.

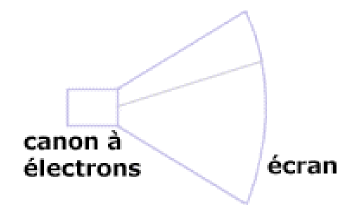
3.14 Périphériques de sortie

3.14.1 L'écran ou le moniteur

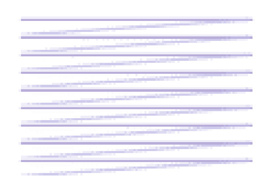
Nous venons de voir une série de périphériques d'entrée, voyons maintenant les périphériques de sortie. Nous en retrouvons principalement deux, l'écran et l'imprimante

L'écran, aussi appelé moniteur, affiche une image dont la netteté dépend de la résolution. Si l'image est composée de petits points, elle sera plus claire. Si les points sont plus gros, elle sera par le fait même beaucoup moins claire. Chacun de ses points s'appelle un pixel.

Les moniteurs (écrans d'ordinateur) sont la plupart du temps des tubes cathodiques, c'est-à-dire un tube en verre dans lequel un canon à électrons émet des électrons dirigés par un champ magnétique vers un écran sur lequel il y a de petits éléments phosphorescents (luminophores) constituant des points (pixels) émettant de la lumière lorsque les électrons viennent les heurter.



Le champ magnétique dévie les électrons de gauche à droite afin de créer un balayage, puis vers le bas une fois arrivé en bout de ligne.

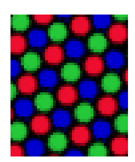


Ce balayage n'est pas perçu par l'oeil humain. Grâce à la persistance rétinienne, essayez par exemple d'agiter votre main devant votre écran pour visualiser ce phénomène. Vous voyez votre main en plusieurs exemplaires.

3.14.2 Le moniteur couleur

Un moniteur noir et blanc permet d'afficher des dégradés de couleur (niveaux de gris) en variant l'intensité du rayon.

Pour les moniteurs couleur, trois faisceaux d'électrons sont utilisés simultanément en visant chacun un point d'une couleur spécifique : un rouge, un vert et un bleu (RGB : Red / Green / Blue ou en français RVB : Rouge / Vert / Bleu). Cependant ces luminophores sont situés de façon tellement proche que l'oeil n'a pas un pouvoir séparateur assez fort. Il voit une couleur composée de ces trois couleurs. Essayez de mettre une minuscule goutte d'eau sur le verre de votre moniteur : celle-ci faisant un effet de loupe va vous faire apparaître les luminophores.



3.14.3 Les moniteurs à cristaux liquides

Cette technologie est basée sur un écran composé de deux plaques transparentes entre lesquelles il y a une fine couche de liquide contenant des molécules (cristaux) capable de s'orienter lorsqu'elles sont soumises à un courant électrique.

Le principal avantage de ce type d'écran est son encombrement réduit, d'où son utilisation sur les ordinateurs portables.

3.14.4 Les caractéristiques

Les moniteurs se caractérisent par les éléments suivants :

- *La définition* : Le nombre de points affiché. Ce nombre de points est actuellement compris entre 640 × 480 (640 points en longueur, 480 points en largeur) et 1600 × 1200.
- *La taille* : Il ne faut pas confondre la définition de l'écran et la taille de l'écran. En effet, un écran d'une taille donnée peut afficher différentes définitions. La taille de l'écran se calcule en mesurant la diagonale de l'écran exprimée en pouces.
- *La résolution* : Elle détermine le nombre de pixels par unité de surface (pixels par pouce carré, en anglais DPI : *Dots Per Inch*).
- *Le pas de masque* : C'est la distance qui sépare deux points. Plus celle-ci est petite, plus l'image est précise.
- *La fréquence de balayage* : C'est le nombre d'images qui sont affichées par seconde. On l'appelle aussi rafraîchissement. Elle est exprimée en hertz (Hz). Plus cette valeur est élevée, meilleur est le confort visuel (on ne voit pas l'image scintiller). Il faut donc qu'elle soit supérieure à 67 hertz (Hz). En dessous de cette limite, l'oeil humain remarque que l'image «clignote».

3.14.5 L'imprimante

Visionner son travail à l'écran est utile mais le résultat final doit souvent se retrouver sur du papier. Il faut alors l'imprimer.

L'imprimante permet de faire une sortie imprimée (sur papier) des données de l'ordinateur.

Il en existe plusieurs types d'imprimantes, dont les plus courantes sont :

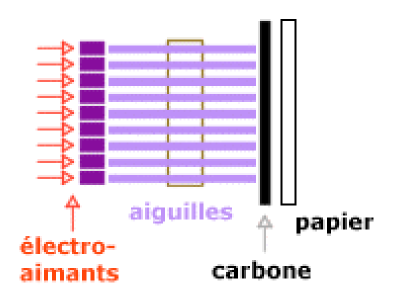
- l'imprimante laser;
- l'imprimante à jet d'encre;
- l'imprimante à bulles d'encre;
- l'imprimante matricielle (à aiguilles);
- l'imprimante à marguerite.

3.14.6 L'imprimante à marguerite

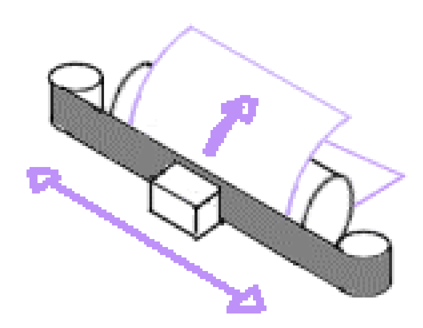
Les imprimantes à marguerite sont basées sur le principe des machines dactylographique. Tous les caractères sont imprimés en relief sur une matrice en forme de marguerite. Pour imprimer, un ruban imbibé d'encre est placé entre la marguerite et la feuille, de telle façon que lorsque la matrice frappe le ruban, celui-ci dépose de l'encre uniquement au niveau du relief du caractère. Ces d'imprimantes sont devenues obsolètes, car elles sont beaucoup trop bruyantes et très peu rapides.

3.14.7 L'imprimante matricielle

Les imprimantes matricielles permettent d'imprimer des documents grâce à un va-et-vient de la tête sur le papier. La tête est constituée de petites aiguilles propulsées par des électro-aimants venant taper contre un ruban de carbone situé entre la tête et le papier.



Ce ruban de carbone défile de façon à ce qu'il y ait continuellement de l'encre dessus.
À chaque fin de ligne un rouleau fait tourner la feuille.



Les imprimantes matricielles les plus récentes sont équipées de têtes d'impression comportant 24 aiguilles, ce qui leur permet d'imprimer avec une résolution de 216 points par pouce.

3.14.8 L'imprimante à jet d'encre

Imprimante équipée d'une tête qui projette de l'encre sur la feuille de papier à travers de petits orifices.

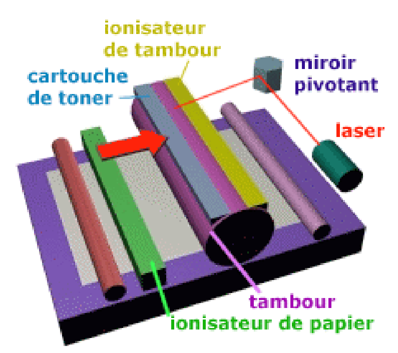
La technologie du jet d'encre a été inventée par Canon. Elle repose sur le principe simple, mais efficace, qu'un fluide chauffé produit des bulles. Le chercheur qui a découvert ce principe avait mis accidentellement en contact une seringue remplie d'encre et un fer à souder, cela créa une bulle dans la seringue qui fit jaillir de l'encre de la seringue.

Les têtes des imprimantes actuelles sont composées de nombreuses buses (jusqu'à 256) équivalentes à plusieurs seringues qui sont chauffées entre 300 et 400 °C plusieurs fois par seconde, grâce à une impulsion électrique.

Chaque buse produit une bulle minuscule qui fait s'éjecter une gouttelette extrêmement fine. Le vide engendré par la baisse de pression aspire une nouvelle goutte...

3.14.9 L'imprimante laser

L'imprimante laser reproduit à l'aide de points l'image reçue du PC. Grâce au laser, les points sont plus petits et la définition est meilleure.



L’imprimante laser fonctionne de la façon suivante : Un ionisateur de papier charge les feuilles positivement. Un ionisateur de tambour charge le tambour négativement. Le laser, pour sa part, grâce à un miroir qui lui permet de se placer, charge le tambour positivement en certains points. Du coup, l'encre du toner chargée négativement se dépose sur les parties du toner ayant été chargées positivement par le laser qui viendront ensuite se déposer sur le papier.

L'imprimante laser, ne possédant pas de tête mécanique, est beaucoup plus rapide et moins bruyante.

3.15 Programmes informatiques

Il faut bien comprendre que l'ordinateur est une machine qui traite rapidement des données sous forme de mots, chiffres, dessins, musique. Mais l'ordinateur n'est pas intelligent, toutes les opérations qu’il exécute doivent être consignées à l'intérieur de programmes. Ce sont les programmes informatiques, c’est-à-dire les logiciels qui lui donnent les instructions à suivre. Ces instructions sont placées sur le disque dur de l’appareil, sur une disquette, un disque compact ou un DVD.

3.16 Les liaisons

3.17 La connexion par la ligne téléphonique

Une ligne téléphonique est conçue pour fonctionner avec un téléphone, c'est pour cela qu'un modem a besoin d'établir une communication avec un ordinateur distant grâce à un numéro de

téléphone avant de pouvoir échanger des informations. On appelle protocole le langage utilisé par les ordinateurs pour communiquer entre eux.

3.17.1 Les modems à 56 Kbit/s

La compagnie Rockwell a présenté une nouvelle norme : la norme K56flex. Cette norme se pose comme alternative à la technologie X2 d'US ROBOTICS. Elle permet d'obtenir des débits de l'ordre de 56Kb/s sur une liaison asynchrone. Elle se différencie par l'encodage et le serveur. Le débit moyen est de 50 Kbps mais la société compte bien arriver à des taux de l'ordre de 110 puis 230 Kbps pour les données offrant un fort taux de compression. Au départ les deux normes étaient sensées pouvoir évoluer.

Depuis 1998 les normes ont été fixées. Ainsi, les modems offrent pour la plupart un bios "flashable" (c'est-à-dire un modem que l'on peut faire évoluer). Grâce à la norme V90, les modems à 56 Kbps devraient maintenant être compatibles entre eux.

3.17.2 Présentation du RNIS

Le Numéris est le réseau téléphonique de France Télécom basé sur la technologie **RNIS** ("*Réseau Numérique à Intégration de Services*", en anglais **ISDN**).

Ce réseau est conçu pour transporter la voix, des données (un avantage entre autres est de pouvoir connaître le numéro de l'appelant, des images, des fax ... D'autre part, la fiabilité et le confort sont incomparable au réseau téléphonique

Depuis novembre 1995, France Télécom a aligné la tarification Numéris sur celles des communications normales. Cependant seules les entreprises semblent avoir accès à cette tarification.

3.17.3 Fonctionnement du RNIS

Il faut avoir un adaptateur pour se connecter sur le réseau Numéris. Le débit est de 64 Kbps (128 en utilisant deux canaux) au lieu de 56 Kbps avec les modems les plus rapides.

3.18 Les lignes spécialisées

Ce sont des lignes louées qui permettent la transmission de données à moyens et hauts débits (2,4 Kbps à 140 Mbps) en liaison point à point ou multipoints (service Transfix).

Les 3 lignes les plus répandues sont les T1 (1.5Mbps), les T2 (6 Mbps), et les T3 (45Mbps). Il existe aussi des lignes nettement plus rapides : ce sont les E1 (2Mbps), E2 (8Mbps), E3 (34Mbps), et E4 (140Mbps) qui sont inaccessibles pour les particuliers.

3.18.1 Quel est le besoin d'une ligne spécialisée?

Pour obtenir une connexion à Internet, il faut, en règle générale,payer un abonnement auprès d'un prestataire Internet ou un service en ligne. Le prix de cette connexion dépend de la vitesse de transfert des données. Il faut choisir celle-ci en fonction du volume de fréquentation du site Web.

3.18.2 Le prix d'une ligne spécialisée

Un site Web ayant une forte fréquentation (environ 10 000 accès par jour soit 50 Mo par jour en moyenne)nécessite une connexion T1 (1.5 Mbps),

Un site moyen (2000 accès par jour) ne nécessite qu'une connexion à 56 Kbps.

Pour les particuliers, il faut de préférence une connexion par câble

3.18.3 La liaison Internet par câble

Les liaisons Internet par câble vous permettent de rester connecté à Internet de façon permanente. Il n'y a plus besoin d'attendre que la connexion s'établisse avec le prestataire, car la connexion avec ce dernier est directe.

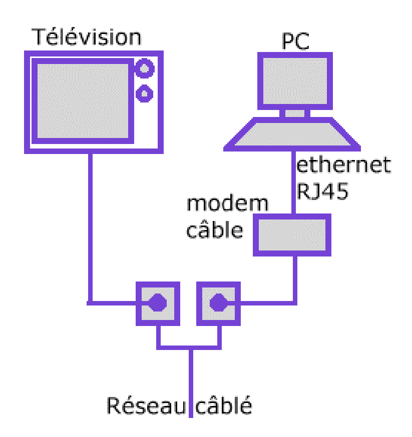
3.18.4 Les avantages

- On ne paye pas la connexion à la minute mais au mois, d'où un coût réduit
- La vitesse est largement supérieure à celle d'un modem ...

3.18.5 Le matériel nécessaire à une liaison par câble

Pour accéder à cette technologie il est nécessaire d'avoir :

- Le câble
- Un fournisseur d'accès par ce câble
- Un modem-câble



Un modem-câble est un appareil qui permet d'accéder à Internet via le réseau de câblodistribution.

Il possède deux types de connexions : une connexion de type coaxial (vers le câble), une connexion de type Ethernet RJ45 (vers la carte réseau de l'ordinateur).

Des vitesses de 10Mbps peuvent être théoriquement atteinte, cependant cette bande passante est partagée suivant l'arborescence qui vous relie à l'opérateur, ainsi il se peut que vous partagiez (et c'est probablement le cas) votre bande-passante avec toutes les personnes de votre immeuble, c'est-à-dire que si tous vos voisins téléchargent des vidéos, les performances ne seront pas au rendez-vous ...

3.18.6 L'ADSL

L'ADSL (*Asymetrical Digital Subscriber Line*) est une technologie de liaison Internet (un peu comme avec un modem standard) dont le débit se situe entre les débits de la ligne de type Numéris et du câble.

Le problème de l'internaute est le taux de transfert des données entre son modem et le central téléphonique. Cette jonction est constituée de fils de cuivre dont on a toujours pensé qu'ils ne pouvaient pas supporter des vitesses de communication de plus de 10 Kb par seconde.

En fait, le réseau téléphonique a été conçu à la base pour transporter des voix, c'est-à-dire que les infrastructures téléphoniques étaient conçues pour utiliser une bande passante de l'ordre de 3,3 Khz.

Pourtant, ces lignes peuvent supporter physiquement des bandes passantes allant jusqu'à 1 Mhz. Il est donc possible, en utilisant les lignes de téléphone, d'optimiser les taux de transfert. Ceux-ci sont fonction de la distance entre l'utilisateur et le central téléphonique, et peuvent s'échelonner entre 1,5 Mbit/s et 10 Mbit/s, offrant des possibilités beaucoup plus grandes que les possibilités actuelles (64 Kbit/s à 128 Kbit/s pour les lignes téléphoniques.

Type de liaison	Taux de transfert théorique
Modem	33.6 Kbit/s
RNIS	64 ou 128 Kbit/s
ADSL	1,5 à 10 Mbit/s en réception 640 Kbit/s en émission
Câble	de 500 Kbit/s à 10 Mbit/s

Il existe différentes technologies basées sur ce principe, elles sont nommées "xDSL" (ADSL,HDSL, SDSL et VDSL); Elles correspondent chacune à une utilisation particulière. C'est l'ADSL qui semble être la plus au point commercialement.

3.18.7 ADSL

Bande haute (1 Mhz) Canal descendant - 8 Mbit/s

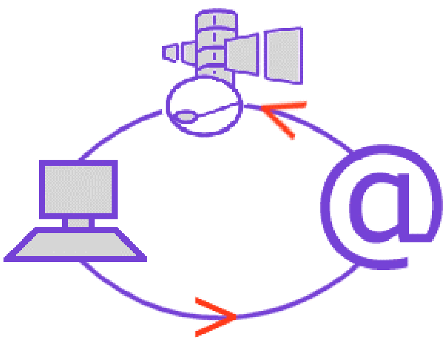
Bande médiane (500 khz) Canal bidirectionnel

Bande basse Téléphonie (0 à 4 Khz)- RNIS (0 à 80 Khz)

3.18.8 La fibre optique

Elle permet de transférer des informations à 100 Mbps, cependant elle coûte très cher et le transfert de voix est difficile, ainsi elle se limite à des réseaux locaux internes aux entreprises.

3.18.9 Le satellite



Effectivement les liaisons satellites sont très rapides, cependant elles sont dans un seul sens (réception), ainsi lorsque l'on veut aller sur un site on ne peut pas en donner l'ordre ... La solution est simple, un modem sur une ligne téléphonique suffit pour envoyer ces informations.

La société Hugues offre déjà un service Internet par satellite, Canal Satellite et TPS sont déjà sur les rangs pour un Internet destiné au grand public.

Le téléchargement de données s'effectue actuellement à un taux de transfert de 400Kbps.

3.18.10 Les ondes hertziennes

Lorsque la construction d'un réseau câblé est trop cher, qu'une zone d'ombre gêne le satellite, le MMDS se révèle être une solution idéale. Il permet de fournir un accès pour une petite ville. Le réseau hertzien est cependant trop encombré pour une couverture nationale.

3.18.11 Le réseau électrique

Une compagnie de téléphone canadienne (Northern Telecom) prétend avoir découvert un nouveau moyen d'accéder à Internet via les lignes électriques.

3.18.12 Le réseau Ethernet

Ethernet (aussi connu sous le nom de *norme IEEE 802.3*) est une technologie de réseau local basé sur le principe que toutes les machines du réseau Ethernet sont connectées à une même ligne de communication, constituée de câbles cylindriques. On distingue différentes variantes de technologies Ethernet suivant le diamètre des câbles utilisés :

- 10Base-2 : Le câble utilisé est un câble coaxial de faible diamètre
- 10Base-5 : Le câble utilisé est un câble coaxial de gros diamètre
- 10Base-T : Le câble utilisé est une paire torsadée, le débit atteint est d'environ 10Mbps
- 100Base-TX : Comme 10Base-T mais avec une vitesse de transmission beaucoup plus importante (100Mbps)

3.18.12.1.1.1.1.1 Technologie	***3.18.12.1.1.1.1.2 Type de câble***	***3.18.12.1.1.1.1.3***	***3.18.12.1.1.1.1.4***
10 Base-2	Câble coaxial de faible diamètre	10 Mb/s	185 m
10 Base-5	Câble coaxial de gros diamètre (0.4 inch)	10 Mb/s	500 m
10 Base-T	Double paire torsadée	10 Mb/s	100 m
100 Base-TX	Double paire torsadée	100 Mb/s	100 m
1000 Base-SX	Fibre optique	1000 Mb/s	500 m

Ethernet est une technologie de réseau très utilisée car le prix de revient d'un tel réseau n'est pas très élevé

Principe de transmission :

Tous les ordinateurs d'un réseau Ethernet sont reliés à une même ligne de transmission, et la communication se fait à l'aide d'un protocole appelé *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detect* ce qui signifie qu'il s'agit d'un protocole d'accès multiple avec surveillance de porteuse (*Carrier Sense*) et détection de collision).

Avec ce protocole toute machine est autorisée à émettre sur la ligne à n'importe quel moment et sans notion de priorité entre les machines. Cette communication se fait de façon simple :

- Chaque machine vérifie qu'il n'y a aucune communication sur la ligne avant d'émettre
- Si deux machines émettent simultanément, alors il y a collision (c'est-à-dire que plusieurs trames de données se trouvent sur la ligne au même moment)
- Les deux machines interrompent leur communication et attendent un délai aléatoire, puis la première ayant passé ce délai peut alors réémettre

Ce principe est basé sur plusieurs contraintes :

- Les paquets de données doivent avoir une taille maximale
- il doit y avoir un temps d'attente entre deux transmissions
- Le temps d'attente varie selon la fréquence des collisions :
- Après la première collision une machine attend une unité de temps
- Après la seconde collision la machine attend deux unités de temps
- Après la troisième collision la machine attend quatre unités de temps
- ... avec bien entendu un petit temps supplémentaire aléatoire

4 Les processus

4.1 Structure des processus

4.1.1 Généralités

Les processus correspondent à l'exécution de tâches : les programmes des utilisateurs, les entrées-sorties… par le système. Un système d'exploitation doit en général traiter plusieurs tâches en même temps. Comme il n'a, la plupart du temps, qu'un processeur, il résout ce problème grâce à un pseudo-parallélisme. Il traite une tâche à la fois, s'interrompt et passe à la suivante. La commutation des tâches étant très rapide, l'ordinateur donne l'illusion d'effectuer un traitement simultané.

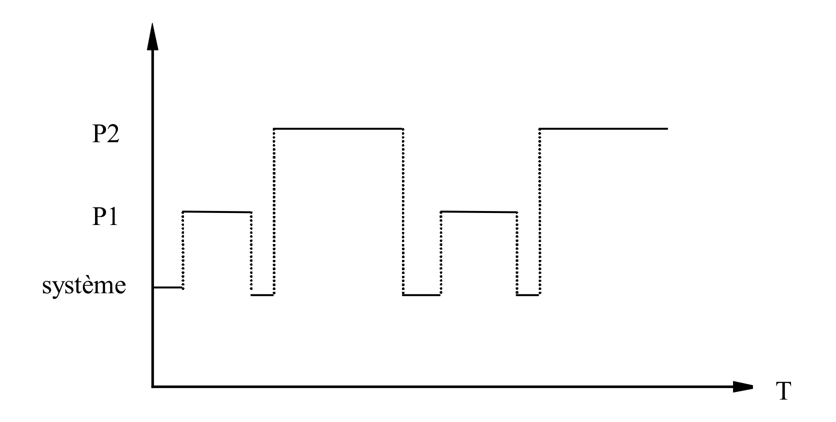


Figure 1 Le multi-tâche

Les processus des utilisateurs sont lancés par un interprète de commande. Ils peuvent eux-mêmes lancer ensuite d'autres processus. On appelle le processus créateur, le père, et les processus créés, les fils. Les processus peuvent donc se structurer sous la forme d'une arborescence. Au lancement du système, il n'existe qu'un seul processus, qui est l'ancêtre de tout les autres.

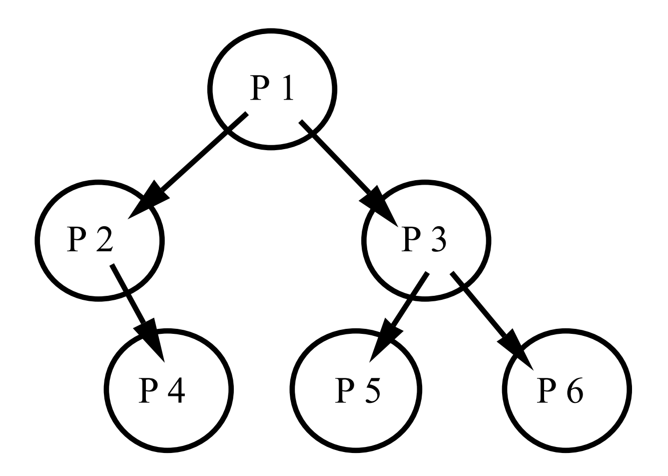


Figure 2 La hiérarchie des processus.

Les processus sont composés d'un espace de travail en mémoire formé de 3 segments : la pile, les données et le code et d'un contexte.

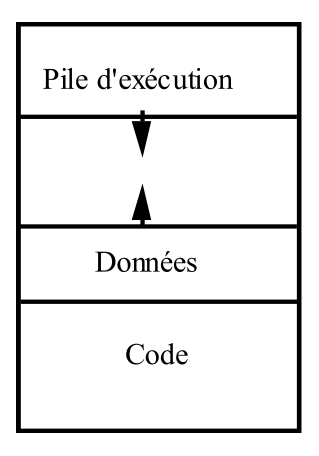


Figure 3 Les segments d'un processus.

Le code correspond aux instructions, en langage d'assemblage, du programme à exécuter. La zone de données contient les variables globales ou statiques du programme ainsi que les allocations dynamiques de mémoire. Enfin, les appels de fonctions, avec leurs paramètres et leurs variables locales, viennent s'empiler sur la pile. Les zones de pile et de données ont des frontières mobiles qui croissent en sens inverse lors de l'exécution du programme. Parfois, on partage la zone de données en données elles-mêmes et en tas. Le tas est alors réservé aux données dynamiques.

Le contexte est formé des données nécessaires à la gestion des processus. Une table contient la liste de tous les processus et chaque entrée conserve leur contexte[1]. Les éléments de la table des processus de Minix, par exemple, ont la forme simplifiée du tableau ci-dessous.

---

[1] Bach, op. cit., p. 158, et Tanenbaum, op. cit., p. 60.

Processus	Mémoire	Fichiers
Registres	Pointeur sur code	Masque umask
Compteur ordinal	Pointeur sur données	Répertoire racine
État du programme	Pointeur sur pile	Répertoire de travail
Pointeur de pile	Statut de fin d'exécution	Descripteurs de fichiers
Date de création	N° de signal du proc. tué	uid effectif
Temps CP utilisé	PID	gid effectif
Temps CP des fils	Processus père	
Date de la proch. alarme	Groupe de processus	
Pointeurs sur messages	uid réel	
Bits signaux en attente	uid effectif	
PID	gid réel	
	gid effectif	
	Bits des signaux	

Tableau 1 Les éléments du contexte de Minix.

Le nombre des emplacements dans la table des processus est limité pour chaque système et pour chaque utilisateur.

La commutation des tâches, le passage d'une tâche à une autre, est réalisée par un ordonnanceur au niveau le plus bas du système. Cet ordonnanceur est activé par des interruptions d'horloge, de disque, de terminaux.

Un processus peut être actif en mémoire centrale (Élu) ou suspendu en attente d'exécution (Prêt). Il peut aussi être Bloqué, en attente de ressource, par exemple au cours d'une lecture de disque. Le diagramme simplifié des états d'un processus est donc :

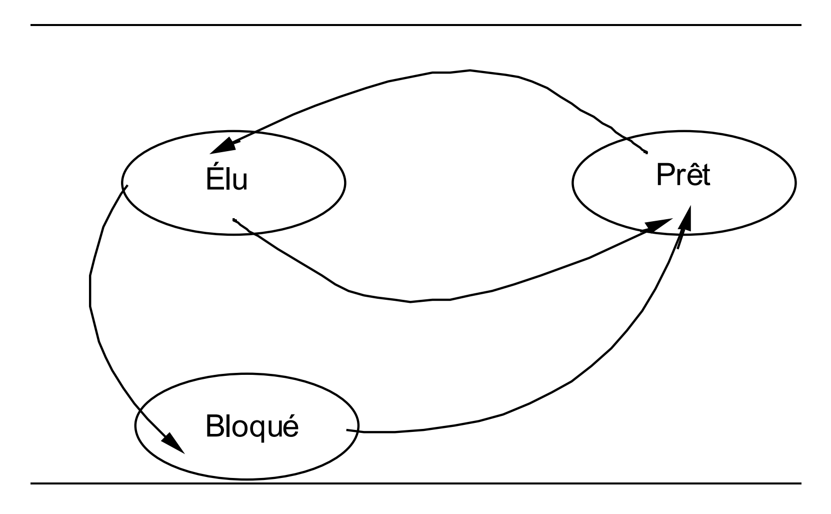


Figure 4 Les états d'un processus.

Le processus passe de l'état élu à l'état prêt et réciproquement au cours d'une intervention de l'ordonnanceur. Cet ordonnanceur se déclenchant, par exemple, sur une interruption d'horloge. Il pourra alors suspendre le processus en cours, s'il a dépassé son quantum de temps, pour élire l'un des processus prêts. La transition de l'état élu à l'état bloqué se produit, par exemple, à l'occasion

d'une lecture sur disque. Ce processus passera de l'état bloqué à l'état prêt lors de la réponse du disque. Ce passage correspond d'une manière générale à la libération d'une ressource.

En fait, sous Unix, l'exécution d'un processus se fait sous deux modes, le mode utilisateur et le mode noyau. Le mode noyau correspond aux appels au code du noyau : `write`, `read`, … Le mode utilisateur correspond aux autres instructions. Un processus en mode noyau ne peut être suspendu par l'ordonnanceur.

Un processus en mode noyau ne peut être suspendu par l'ordonnanceur. Il passe à l'état préempté à la fin de son exécution dans ce mode – à moins d'être bloqué ou détruit –. Cet état est virtuellement le même que prêt en mémoire.

Par ailleurs, lorsque la mémoire ne peut contenir tous les processus prêts, un certain nombre d'entre eux sont déplacés sur le disque. Un processus peut alors être prêt en mémoire ou prêt sur le disque. De la même manière, on a des processus bloqués en mémoire ou bien bloqués sur disque.

Enfin, les processus terminés ne sont pas immédiatement éliminés de la tables des processus – ils ne le sont que par une instruction explicite de leur père –. Ceci correspond à l'état défunt.

Les états d'un processus Unix[2] sont alors :

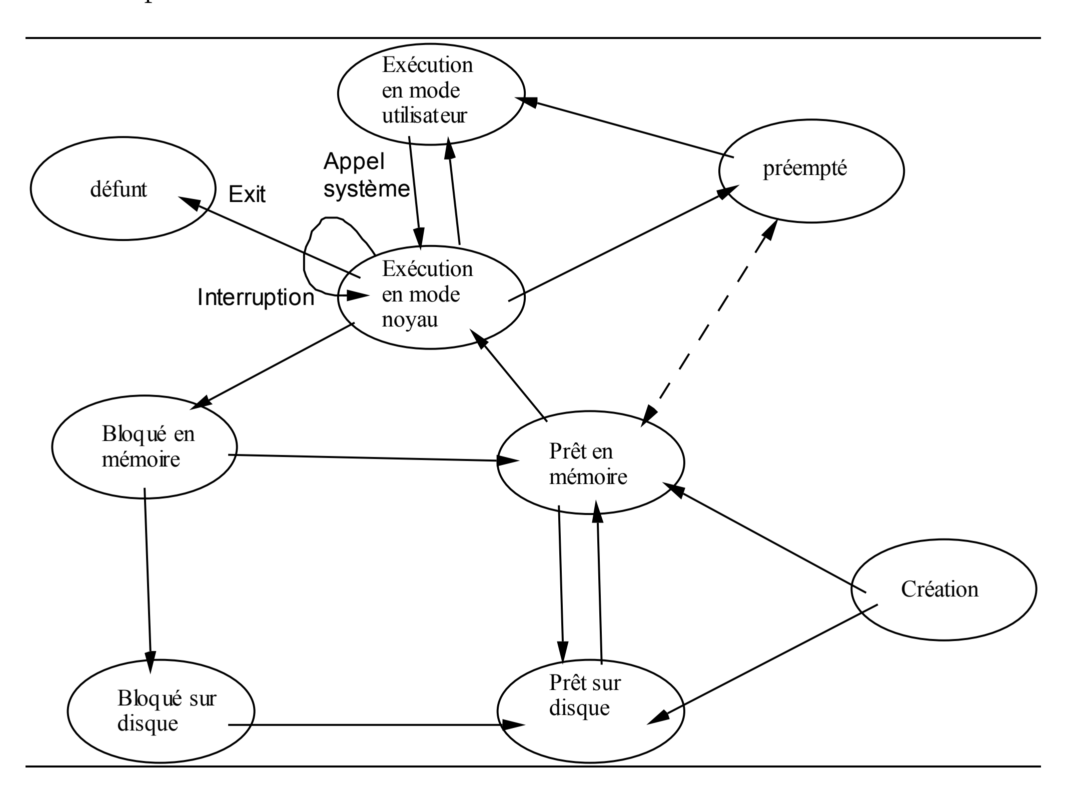


Figure 5 Les états des processus d'Unix.

[2]Bach, op.cit., p. 156.

4.1.2 Les processus sous Unix

4.1.2.1 Les fonctions fondamentales

Un processus peut se dupliquer – et donc ajouter un nouveau processus – par la fonction : `fork(void)` qui retourne `-1` en cas d'échec. En cas de réussite, la fonction retourne `0` dans le processus fils et le n° du processus fils – *Process Identifier* ou PID – dans le père. Cette fonction transmet une partie du contexte du père au fils : les descripteurs des fichiers standards et des autres fichiers, les redirections… Les deux programmes sont sensibles aux mêmes interruptions. À l'issue d'un `fork()` les deux processus s'exécutent simultanément.

La création d'un nouveau processus ne peut se faire que par le recouvrement du code d'un processus existant grâce à la fonction :

```
int execl(char *ref, char *arg0, char *argn, 0)
```

`ref` est une chaîne de caractères donnant l'adresse du nouveau programme à substituer et à exécuter. `arg0`, `arg1`, …, `argn` sont les arguments du programme. Le premier argument, `arg0`, reprend en fait le nom du programme.

Les fonctions `execle()` et `execlp()` ont des arguments et des effets semblables. Elles transmettent en plus respectivement la variable chemin (`PATH`) ou les variables d'environnement au complet.

Les fonctions `int execv(char *ref, char *argv[])` ainsi que `execve()` et `execvp()` sont des variantes analogues. Les paramètres du programme à lancer sont transmis dans le tableau de chaînes : `argv[][]`. Ce tableau est semblable aux paramètres de la fonction principale d'un programme C : `main(int argc, char **argv)`.

Ces fonctions rendent `-1` en cas d'échec. Par exemple :

```
void main()
{ execl("/bin/ls", "ls", "-l", (char *) 0);
/* avec execlp, le premier argument peut n'être que ls et
non /bin/ls */
   printf("pas d'impression\n");
}
```

La création d'un nouveau processus – l'occupation d'un nouvel emplacement de la table des processus – implique donc la duplication d'un processus existant. Ce dernier sera le père. On substituera ensuite le code du fils par le code qu'on désire exécuter. Au départ, le processus initial lance tous les processus utiles au système.

Un processus se termine lorsqu'il n'a plus d'instructions ou lorsqu'il exécute la fonction :

```
void exit(int statut)
```

L'élimination d'un processus terminé de la table ne peut se faire que par son père, grâce à la fonction :

```
int wait(int *code_de_sortie)
```

Avec cette instruction, le père se bloque en attente de la fin d'un fils. Elle rendra le n° PID du premier fils mort trouvé. La valeur du code de sortie est reliée au paramètre d'`exit` de ce fils. On peut donc utiliser l'instruction `wait` pour connaître la valeur éventuelle de retour, fournie par `exit()`, d'un processus. Ce mode d'utilisation est analogue à celui d'une fonction. `wait()` rend `-1` en cas d'erreur.

Si le fils se termine sans que son père l'attende, le fils passe à l'état `defunct` dans la table. Si, au contraire, le père se termine avant le fils, ce dernier est rattaché au processus initial. Le processus initial passe la plupart de son temps à attendre les processus orphelins.

Grâce aux 3 instructions, `fork()`, `exec()`, et `wait()`, on peut écrire un interprète de commandes simplifié. Il prend la forme suivante :

```
while(1) {/* attend commande */
   lire_commande(commande, paramètres);
   if (fork() != 0) {
        wait(&statut);           /* proc. père */
   } else {
        execv(commande, paramètres);  /* proc. fils */
   }
}
```

Les fonctions Unix rendent `-1` en cas d'erreur. Dans ce cas, elles positionnent la variable entière globale `errno`. On peut imprimer cette valeur d'erreur et constater la cause qui est décrite dans le fichier `errno.h`. On peut aussi imprimer le texte en clair avec la fonction `perror(char*)`. Dans le cas d'une programmation professionnelle, il est indispensable de vérifier la valeur de retour de toutes les fonctions faisant appel au système et de se protéger des erreurs par un codage du type :

```
if ((retour = fonction()) == -1) {
     perror("plantage");
     exit(0);}
```

4.1.2.2 Les signaux

L'instruction : `int kill(int pid, int signal)`

permet à un processus d'envoyer un signal à un autre processus. `pid` est le n° du processus à détruire et `signal`, le n° du signal employé. La fonction `kill` correspond à des interruptions logicielles.

Par défaut, un signal provoque la destruction du processus récepteur, à condition bien sûr, que le processus émetteur possède ce droit de destruction.

La liste des valeurs de `signal` comprend, entre autres :

Nom du signal	N° du signal	Commentaires
SIGUP	1	signal émis lors d'une déconnexion
SIGINT	2	^C
SIGQUIT	3	^\
SIGKILL	9	signal d'interruption radicale
SIGALRM	14	signal émis par `alarm(int sec)` au bout de sec secondes
SIGCLD	20	signal émis par un fils qui se termine, à son père

Tableau 2 Quelques signaux d'Unix.

Les valeurs des signaux dépendent de l'implantation. La liste complète est définie par des macros dans `signal.h`

`kill()` rend 0 en cas de succès et `-1` en cas d'échec.

Un processus peut détourner les signaux reçus et modifier son comportement par l'appel de la fonction : `void (*signal(int signal, void (*fonction)(int)))`

avec `fonction` pouvant prendre les valeurs:

Nom	Action
SIG_IGN	le processus ignorera l'interruption correspondante
SIG_DFL	le processus rétablira son comportement par défaut lors de l'arrivée de l'interruption (la terminaison)
void fonction(int n)	Le processus exécutera `fonction`, définie par l'utilisateur, à l'arrivée de l'interruption `n`. Il reprendra ensuite au point où il a été interrompu

Tableau 3 Les routines d'interruption d'Unix.

L'appel de la fonction `signal` positionne l'état des bits de signaux dans la table des processus.

Par ailleurs, la fonction `signal(SIGCLD, SIG_IGN)`, permet à un père d'ignorer le retour de ses fils sans que ces derniers encombrent la table des processus à l'état `defunct`.

4.1.2.3 Quelques fonctions d'identification d'Unix

La commande Unix `ps -ef`, donne la liste des processus en activité sur le système. Chaque processus possède un identificateur et un groupe. On les obtient par les fonctions :

`int getpid(void)`et

`int getpgrp(void)`

Les fils héritent du groupe de processus du père. On peut changer ce groupe par la fonction :

`int setpgrp(void)`

Chaque processus possède, d'autre part, un utilisateur réel et effectif. On les obtient respectivement par les fonctions :

`int getuid(void)`et

`int geteuid(void)`

L'utilisateur réel est l'utilisateur ayant lancé le processus. Le numéro d'utilisateur effectif sera utilisé pour vérifier certains droits d'accès (fichiers et envoi de signaux). Il correspond normalement au numéro d'utilisateur réel. On peut cependant positionner l'utilisateur effectif d'un processus au propriétaire du fichier exécutable. Ce fichier s'exécutera alors avec les droits du propriétaire et non avec les droits de celui qui l'a lancé. La fonction :

`int setuid(int euid)`

permet de commuter ces numéros d'utilisateur de l'un à l'autre : réel à effectif et vice-versa. Elle rend `0` en cas de succès.

4.2 Communication entre les processus

Les processus peuvent communiquer par l'intermédiaire de fichiers. Néanmoins, certains systèmes d'exploitation fournissent des mécanismes permettant les communications directes.

L'interprète de commandes Unix utilise le signe « `|` » qu'il relie à l'appel système `pipe()`. Il permet à deux processus de s'exécuter en même temps. Le premier fournissant des données que le second exploite au fur et à mesure de leur production.

4.2.1 Les tubes de communication avec Unix

Le système Unix offre des primitives qui permettent de synchroniser facilement des processus lecteurs et écrivains dans un tampon sans passer par des sémaphores. L'appel de la fonction du noyau `pipe()` crée un tampon de données, un tube, dans lequel deux processus pourront venir respectivement lire et écrire. La fonction fournit, d'autre part, deux descripteurs, analogues aux descripteurs de fichiers, qui serviront de référence pour les opérations de lecture et d'écriture. Le système réalise la synchronisation de ces opérations de manière interne.

Les deux processus communicants doivent partager les mêmes descripteurs obtenus par `pipe()`. Il est donc commode qu'ils aient un ancêtre commun car les descripteurs de fichiers sont hérités. Dans l'exemple suivant, deux processus, un père et un fils communiquent par l'intermédiaire d'un tube. Le père écrit les lettres de l'alphabet que lit le fils :

4.2.2 Les messages

Les messages forment un mode de communication privilégié entre les processus. Ils sont utilisés dans le système pédagogique Minix de Tanenbaum. Ils sont au cœur de Mach qu'on présente comme un successeur possible d'Unix et qui a inspiré Windows NT pour certaines parties. Par ailleurs, ils s'adaptent très bien à une architecture répartie et leur mode de fonctionnement est voisin des échanges de données sur un réseau.

Les communications de messages se font à travers deux opérations fondamentales : `envoie(message)` et `reçois(message)`. (`send` et `receive`). Les messages sont de tailles variables ou fixes. Les opérations d'envoi et de réception peuvent être soit directes entre les processus, soit indirectes par l'intermédiaire d'une boîte aux lettres.

Les communications directes doivent identifier le processus, par un n° et une machine, par exemple. On aura alors : `envoie(Proc, Message)` et `reçois(Proc, Message)`. Dans ce cas la communication est établie entre deux processus uniquement, par un lien relativement rigide et bidirectionnel. On peut rendre les liaisons plus souples en laissant vide l'identité du processus dans la fonction `reçois`.

Les communications peuvent être indirectes grâce à l'utilisation d'une boîte aux lettres (un « port » dans la terminologie des réseaux). Les liens peuvent alors unir plus de deux processus du moment qu'ils partagent la même boîte aux lettres. On devra néanmoins résoudre un certain nombre de problèmes qui peuvent se poser, par exemple, si deux processus essayent de recevoir simultanément le contenu d'une même boîte.

Les communications se font de manière synchrone ou asynchrone. Le synchronisme peut se représenter par la capacité d'un tampon de réception. Si le tampon n'a pas de capacité, l'émetteur doit attendre que le récepteur lise le message pour pouvoir continuer. Les deux processus se

synchronisent sur ce transfert et on parle alors d'un « rendez-vous ». Deux processus asynchrones : P et Q, peuvent aussi communiquer de cette manière en mettant en œuvre un mécanisme d'acquittement :

P	Q
`envoie(Q, message)`	`reçois(P, message)`
`reçois(Q, message)`	`envoie(P, acquittement)`

Tableau 4 Les fonctions de messages.

4.2.3 La mémoire partagée

On peut concevoir des applications qui communiquent à travers un segment de mémoire partagée. Le principe est le même que pour un échange d'informations entre deux processus par un fichier. Dans le cas d'une zone de mémoire partagée, on devra déclarer une zone commune par une fonction spécifique, car la zone mémoire d'un processus est protégée.

Le système Unix fournit les primitives permettant de partager la mémoire. NT aussi sous le nom de fichiers mappés en mémoire. Ces mécanismes, bien que très rapides, présentent l'inconvénient d'être difficilement adaptables aux réseaux. Pour les communications locales, la vitesse est sans doute semblable à celle de la communication par un fichier à cause de la mémoire cache. Lorsqu'il a besoin de partager un espace mémoire, le programmeur préférera utiliser des fils d'exécution.

4.3 Ordonnancement

L'ordonnancement règle les transitions d'un état à un autre des différents processus. Cet ordonnancement a pour objectifs de :

1. Maximiser l'utilisation du processeur;
2. Être équitable entre les différents processus;
3. Présenter un temps de réponse acceptable;
4. Avoir un bon rendement;
5. Assurer certaines priorités.

4.3.1 Le tourniquet

Cet algorithme est l'un des plus utilisés et l'un des plus fiables. Chaque processus prêt dispose d'un quantum de temps pendant lequel il s'exécute. Lorsqu'il a épuisé ce temps ou qu'il se

bloque, par exemple sur une entrée-sortie, le processus suivant de la file d'attente est élu et le remplace. Le processus suspendu est mis en queue du tourniquet.

Le seul paramètre important à régler, pour le tourniquet, est la durée du quantum. Il doit minimiser le temps de gestion du système et cependant être acceptable pour les utilisateurs. La part de gestion du système correspond au rapport de la durée de commutation sur la durée du quantum. Plus le quantum est long plus cette part est faible, mais plus les utilisateurs attendent longtemps leur tour. Un compromis peut se situer, suivant les machines, de 100 à 200 ms.

4.3.2 Les priorités

Dans l'algorithme du tourniquet, les quanta égaux rendent les différents processus égaux. Il est parfois nécessaire de privilégier certains processus par rapport à d'autres. L'algorithme de priorité choisit le processus prêt de plus haute priorité.

Ces priorités peuvent être statiques ou dynamiques. Les processus du système auront des priorités statiques (non-modifiables) fortes. Les processus des utilisateurs verront leurs priorités modifiées, au cours de leur exécution, par l'ordonnanceur. Ainsi un processus qui vient de s'exécuter verra sa priorité baisser. Pour un exemple d'exécution avec priorités, on pourra consulter Bach[3].

4.3.3 Le tourniquet avec priorités

On utilise généralement une combinaison des deux techniques précédentes. À chaque niveau de priorité correspond un tourniquet. L'ordonnanceur choisit le tourniquet non vide de priorité la plus forte et l'exécute.

Pour que tous les processus puissent s'exécuter, il est nécessaire d'ajuster périodiquement les différentes priorités.

4.3.4 L'ordonnancement des fils d'exécution

Les fils d'exécution sont soumis à un ordonnancement. Dans Windows NT, les fils d'exécution ont 32 niveaux de priorité qui sont soit fixes soit dynamiques. La priorité la plus haute est toujours celle qui s'exécute. Dans le cas d'une priorité dynamique, les valeurs de cette priorité varient entre deux bornes. Elle augmente, par exemple, lors d'une attente d'entrée-sortie. On peut changer la priorité des fils d'exécution dans Windows NT.

---

[3] op. cit., p. 267-270.

5 La mémoire

5.1 Introduction

La mémoire principale est le lieu où se trouvent les programmes et les données quand le processeur les exécute. On l'oppose au concept de mémoire secondaire, représentée par les disques, de plus grande capacité, où les processus peuvent séjourner avant d'être exécutés.

De manière encore plus vive que pour les autres ressources informatiques, le prix des mémoires a baissé et la capacité unitaire des circuits a augmenté. Cependant la nécessité de la gérer de manière optimale est toujours fondamentale, car en dépit de sa grande disponibilité, elle n'est, en général, jamais suffisante. Ceci en raison de la taille continuellement grandissante des programmes.

5.1.1 La multiprogrammation

Le concept de multiprogrammation s'oppose à celui de monoprogrammation. La monoprogrammation ne permet qu'à un seul processus utilisateur d'être exécuté. Cette technique n'est plus utilisée que dans les micro-ordinateurs. On trouve alors en mémoire, par exemple dans le cas de MS-DOS : le système en mémoire basse, les pilotes de périphériques en mémoire haute (dans une zone allant de 640 ko à 1 Mo) et un programme utilisateur entre les deux.

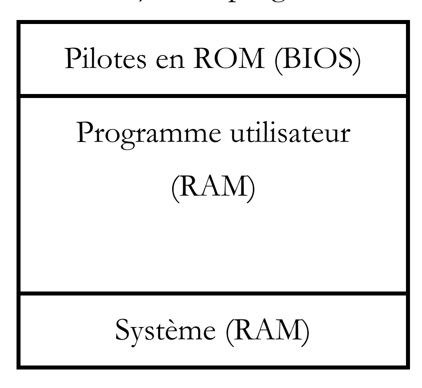


Figure 6 L'organisation de la mémoire du DOS.

La multiprogrammation autorise l'exécution de plusieurs processus indépendant à la fois[4]. Cette technique permet d'optimiser le taux d'utilisation du processeur en réduisant notamment les attentes sur des entrées-sorties. La multiprogrammation implique le séjour de plusieurs

[4] Il est à noter que MS-DOS, bien qu'il ne soit pas multiprogrammable, n'est pas non plus un système monoprogrammé *stricto sensu*; c'est une créature qui tient des deux.

programmes en même temps en mémoire et c'est cette technique qui a donné naissance à la gestion moderne de la mémoire.

5.1.2 Les registres matériels

La gestion de la mémoire est presque impossible sans l'aide du matériel. Celui-ci doit notamment assurer la protection. Dans les systèmes multi-utilisateurs, par exemple, on doit interdire à un utilisateur d'accéder n'importe comment au noyau du système ou aux autres programmes des utilisateurs.

Pour assurer une protection fondamentale, on dispose, sur la plupart des processeurs[5], de deux registres délimitant le domaine d'un processus : le registre de base et le registre de limite. La protection est alors assurée par le matériel qui compare les adresses émises par le processus à ces deux registres.

5.2 Concepts fondamentaux

5.2.1 Production d'un programme

Avant d'être exécuté, un programme doit passer par plusieurs étapes. Au début, le programmeur crée un fichier et écrit son programme dans un langage source, le C par exemple. Un **compilateur** transforme ce programme en un module objet. Le module objet représente la traduction des instructions en C, en langage machine. Le code produit est en général relogeable, commençant à l'adresse 00000 et pouvant se translater à n'importe quel endroit de la mémoire en lui donnant comme référence initiale le registre de base. Les adresses représentent alors le décalage par rapport à ce registre.

5.2.2 Principes de gestion

Pour effectuer le chargement, le système **alloue** un espace de mémoire libre et il y place le processus. Il libérera cet emplacement une fois le programme terminé.

Dans beaucoup de cas, il n'est pas possible de faire tenir tous les programmes ensemble en mémoire. Parfois même, la taille d'un seul programme est trop importante. Le programmeur peut, dans ce cas, mettre en œuvre une stratégie de recouvrement (*overlay*) consistant à découper un programme important en modules et à charger ces modules quand ils sont nécessaires. Ceci est cependant très fastidieux et nécessite, pour chaque nouveau programme, un redécoupage.

---

[5] Malheureusement pas sur le 8086 ce qui a eu des conséquences considérables sur les systèmes DOS et Windows.

Les systèmes d'exploitation modernes mettent en œuvre des stratégies qui libèrent le programmeur de ces préoccupations. Il existe deux stratégies principales pour gérer les chargements : le **va-et-vient** et la **mémoire virtuelle**.

5.3 L'allocation

Avant d'implanter une technique de gestion de la mémoire centrale par va-et-vient, il est nécessaire de connaître son état : les zones libres et occupées; de disposer d'une stratégie d'allocation et enfin de procédures de libération. Les techniques que nous allons décrire servent de base au va-et-vient; on les met aussi en œuvre dans le cas de la multiprogrammation simple où plusieurs processus sont chargés en mémoire et conservés jusqu'à la fin de leur exécution.

5.3.1 État de la mémoire

Le système garde la trace des emplacements occupés de la mémoire par l'intermédiaire d'une table de bits ou bien d'une liste chaînée. La mémoire étant découpée en unités, en blocs, d'allocation.

5.3.1.1 Tables de bits

On peut conserver l'état des blocs de mémoire grâce à une table de bits. Les unités libres étant notées par 0 et ceux occupées par un 1. (ou l'inverse).

0	0	1	1	0	0	

Figure 7

La technique des tables de bits est simple à implanter, mais elle est peu utilisée. On peut faire la remarque suivante : plus l'unité d'allocation est petite, moins on a de pertes lors des allocations, mais en revanche, plus cette table occupe de place en mémoire.

5.3.1.2 Listes chaînées

On peut représenter la mémoire par une liste chaînée de structures dont les membres sont : le type (libre ou occupé), l'adresse de début, la longueur, et un pointeur sur l'élément suivant.

Pour une mémoire ayant l'état suivant :

0 5 8 10 15 20

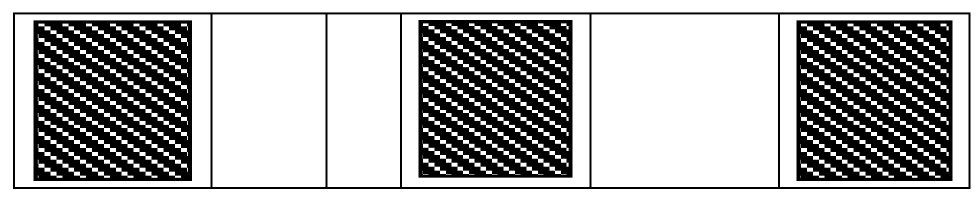


Figure 8

on aurait la liste :

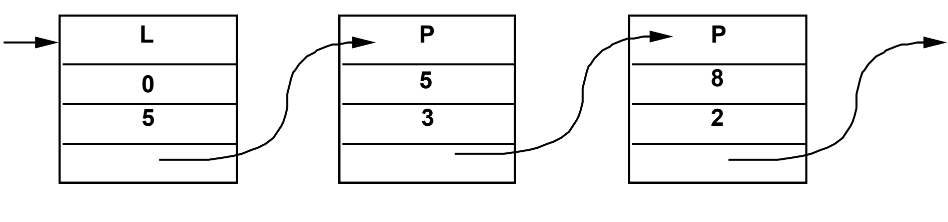


Figure 9

On peut légèrement modifier ce schéma en prenant deux listes : l'une pour les processus et l'autre pour les zones libres. La liste des blocs libres peut elle-même se représenter en réservant quelques octets de chaque bloc libre pour contenir un pointeur sur le bloc libre suivant.

Le système MS-DOS utilise une variante de ce procédé grâce à un en-tête de 16 octets qui précède chaque zone (arène) en mémoire. Les en-têtes contiennent notamment le type de l'arène (un pointeur sur le contexte du processus[6] ou 0) et sa taille.

5.3.2 Politiques d'allocation

L'allocation d'un espace libre pour un processus peut se faire suivant trois stratégies principales : le « premier ajustement » (*first fit*), le « meilleur ajustement » (*best fit*), et le « pire ajustement » (*worst fit*).

Dans le cas du « premier ajustement », on prend le premier bloc libre de la liste qui peut contenir le processus qu'on désire charger. Le « meilleur ajustement » tente d'allouer au processus l'espace mémoire le plus petit qui puisse le contenir. Le « pire ajustement » lui prend le plus grand bloc disponible et le fragmente en deux.

Des simulations ont montré que le « premier ajustement » était meilleur que les autres. Paradoxalement, le « meilleur ajustement », qui est plus coûteux, n'est pas optimal car il produit une fragmentation importante.

5.3.3 Libération

La libération se produit quand un processus est évacué de la mémoire. On marque alors le bloc à libre et on le fusionne éventuellement avec des blocs adjacents.

Supposons que X soit le bloc qui se libère, on a les schémas de fusion suivants :

<table>
<tr><th colspan="3">Avant</th><th colspan="3">Après</th></tr>
<tr><td>A</td><td>X</td><td>B</td><td>A</td><td>L</td><td>B</td></tr>
<tr><td>A</td><td>X</td><td>L</td><td>A</td><td colspan="2">L</td></tr>
<tr><td>L</td><td>X</td><td>B</td><td colspan="2">L</td><td>B</td></tr>
<tr><td>L</td><td>X</td><td>L</td><td colspan="3">L</td></tr>
</table>

Figure 10

5.3.4 La récupération de mémoire

La fragmentation de la mémoire est particulièrement dommageable car elle peut saturer l'espace disponible rapidement. Ceci est particulièrement vrai pour les gestionnaires de fenêtrage. Pour la diminuer, on peut compacter régulièrement la mémoire. Pour cela, on déplace les processus, par exemple, vers la bas de la mémoire et on les range l'un après l'autre de manière contiguë. On construit alors un seul bloc libre dans le haut de la mémoire. Cette opération est coûteuse et nécessite parfois des circuits spéciaux.

Par ailleurs, une fois qu'un objet ou une zone a été utilisé, le programmeur système doit récupère la mémoire « à la main » par une libération du pointeur sur cette zone – `free()`. Ceci est une source d'erreurs car on oublie parfois cette opération. Si on alloue dans une fonction, il n'y a plus moyen d'accéder au pointeur après le retour de la fonction. Le bloc de mémoire est alors perdu et inutilisable. On a une « fuite de mémoire » – *memory leak*.

Certains langages ou systèmes incorporent la récupération automatique de mémoire – *garbage collection*. Ils libèrent ainsi le programmeur de cette tâche. C'est le cas de Java. Il est alors très facile d'éliminer immédiatement une zone par l'affectation `objet = null`. Sans ça, le récupérateur doit déterminer tout seul qu'une zone n'a plus de référence dans la suite du programme.

5.4 Le va-et-vient

[6] *Program Segment Prefix* ou PSP.

Le va-et-vient est mis en œuvre lorsque tous les processus ne peuvent pas tenir simultanément en mémoire. On doit alors en déplacer temporairement certains sur une mémoire provisoire, en général, une partie réservée du disque (*swap area* ou *backing store*).

Sur le disque, la zone de va-et-vient d'un processus peut être allouée à la demande dans la zone de va-et-vient générale (*swap area*). Quand un processus est déchargé de la mémoire centrale, on lui recherche une place. Les places de va-et-vient sont gérées de la même manière que la mémoire centrale. La zone de va-et-vient d'un processus peut aussi être allouée une fois pour toute au début de l'exécution. Lors du déchargement, le processus est sûr d'avoir une zone d'attente libre sur le disque.

Le système exécute pendant un certain quantum de temps[7] les processus en mémoire puis déplace un de ces processus au profit d'un de ceux en attente dans la mémoire provisoire. L'algorithme de remplacement peut être le tourniquet.

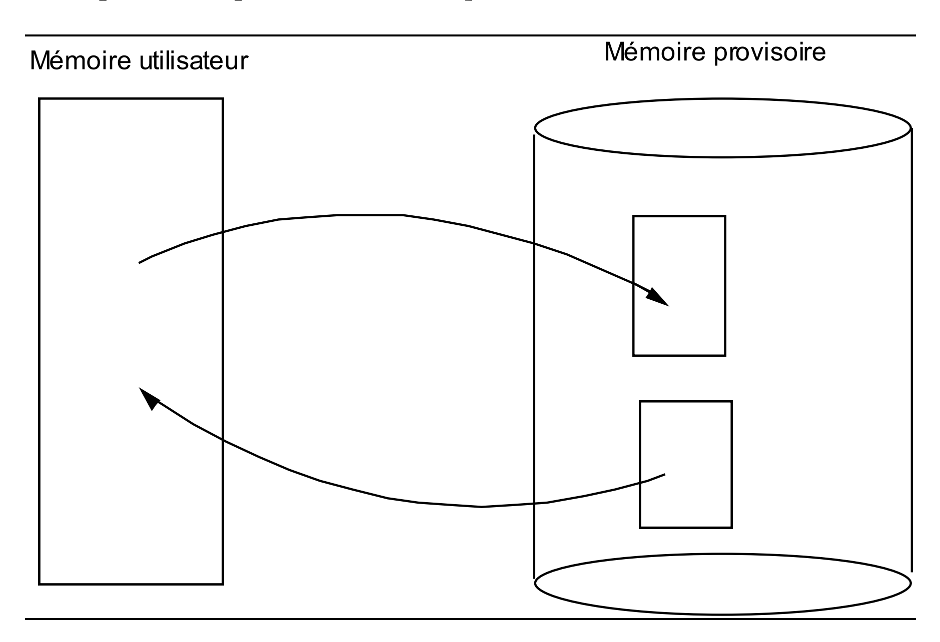


Figure 11

Le système de va-et-vient, s'il permet de pallier le manque de mémoire nécessaire à plusieurs utilisateurs, n'autorise cependant pas l'exécution de programmes de taille supérieure à celle de la mémoire centrale.

5.5 La pagination

[7] Ce quantum est bien sûr plus long que le quantum de commutation de tâches vu au chapitre sur les processus. Il tient compte du fait que les temps de chargement et de déchargement sont beaucoup plus longs que les temps de commutation de processus.

La pagination permet d'avoir en mémoire un processus dont les adresses sont non contiguës. Pour réaliser ceci, on partage l'espace d'adressage du processus et la mémoire physique en pages de quelques kilo-octets. Les pages du processus sont chargées à des pages libres de la mémoire.

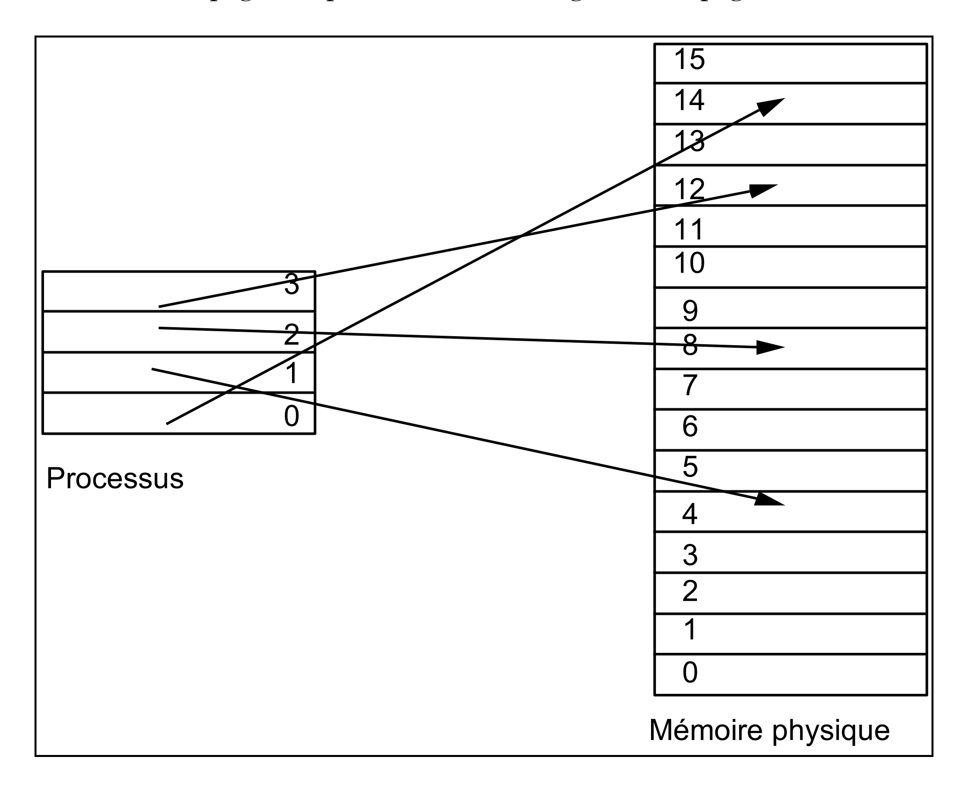


Figure 12

On conserve l'emplacement des pages par une table de transcodage :

0	14
1	4
2	8
3	12

La pagination permet d'écrire des programmes ré-entrants, c'est à dire où certaines pages de codes sont partagées par plusieurs processus.

5.6 La segmentation

Alors que la pagination propose un espace d'adressage plat et indifférencié, (ceci est offert par la famille de μ-processeurs 68000), la segmentation partage les processus en segments bien spécifiques. On peut ainsi avoir des segments pour des procédures, pour la table de symboles, pour le programme principal, etc. Ces segments peuvent être relogeables et avoir pour origine un registre de base propre au segment. La segmentation permet aussi le partage, par exemple du code d'un éditeur entre plusieurs processus. Ce partage porte alors sur un ou plusieurs segments.

Un exemple réduit d'architecture segmentée est donné par la famille 8086 qui possède 3 segments : le code (Code Segment), la pile (Stack Segment) et les données (Data Segment).

5.7 La mémoire virtuelle

5.7.1 Présentation

La mémoire virtuelle permet d'exécuter des programmes dont la taille excède la taille de la mémoire réelle. Pour ceci, on découpe (on « pagine ») les processus ainsi que la mémoire réelle en pages de quelques kilo-octets (1, 2 ou 4 ko généralement).

L'encombrement total du processus constitue l'espace d'adressage ou la mémoire virtuelle. Cette mémoire virtuelle réside sur le disque. À la différence de la pagination présentée précédemment, on ne charge qu'un sous-ensemble de pages en mémoire. Ce sous-ensemble est appelé l'espace physique (réel).

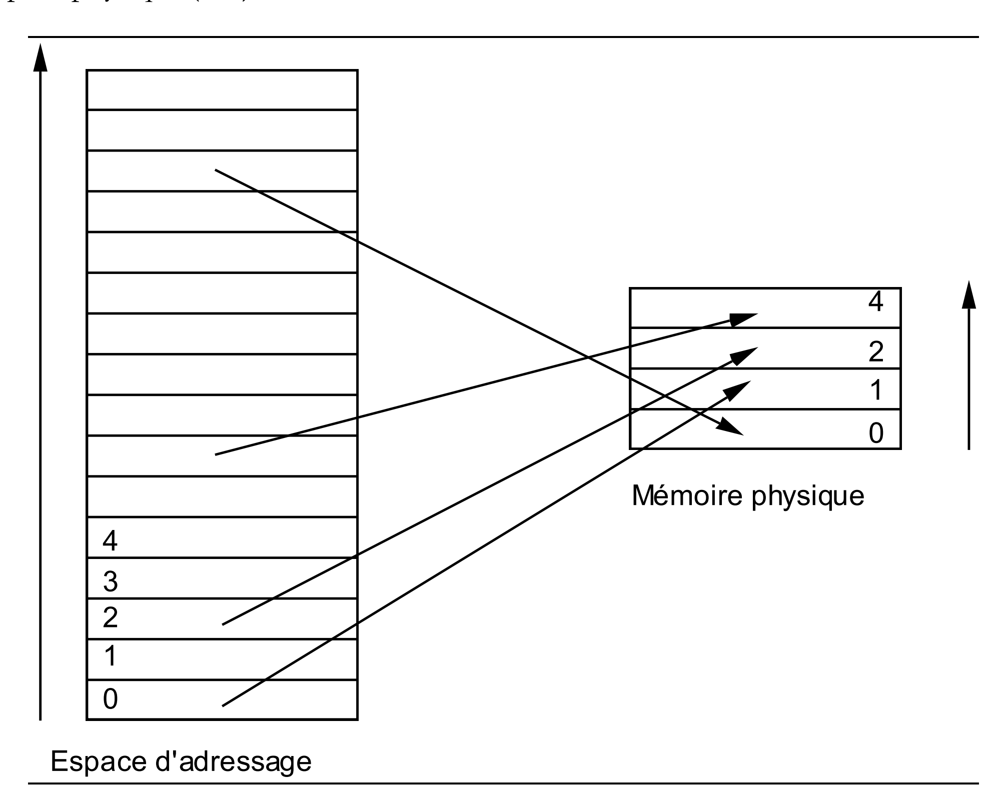


Figure 13

Lorsqu'une adresse est générée, elle est transcodée, grâce à une table, pour lui faire correspondre son équivalent en mémoire physique. Ce transcodage est effectué par des circuits matériels de gestion : *Memory Management Unit* (MMU). Si cette adresse correspond à une adresse en mémoire physique, le MMU transmet sur le bus l'adresse réelle, sinon il se produit un **défaut de page**. Pour pouvoir accéder à la page dont on a généré l'adresse, on devra préalablement la

charger en mémoire réelle. Pour cela, on choisit parmi les pages réelles une page « victime »; si cette dernière a été modifiée, on la reporte en mémoire virtuelle (sur le disque) et on charge à sa place la page à laquelle on désirait accéder.

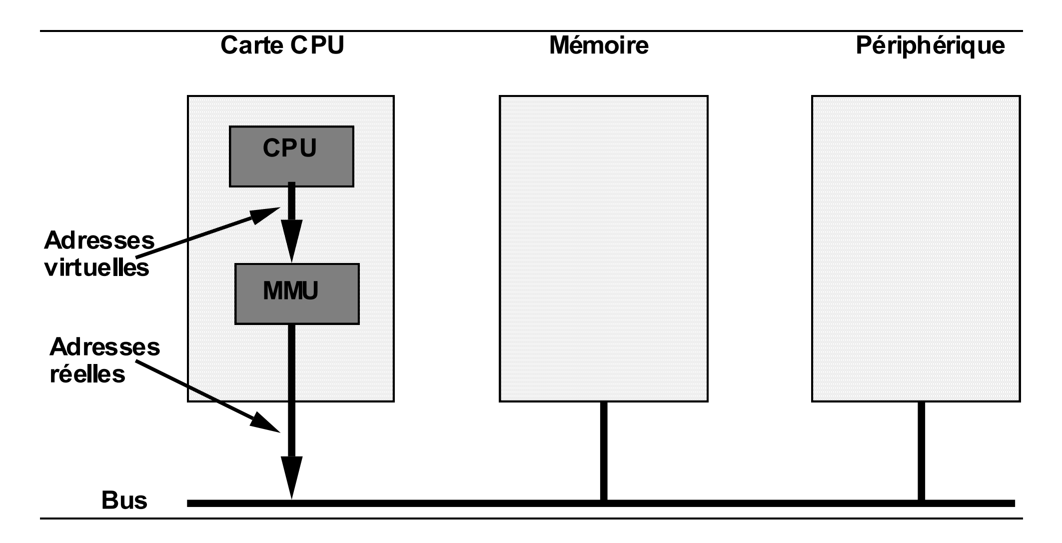


Figure 14 L'unité de gestion de mémoire.

5.7.2 Algorithmes de remplacement de pages

L'algorithme de remplacement de page optimal consiste à choisir comme victime, la page qui sera appelée le plus tard possible. On ne peut malheureusement pas implanter cet algorithme, mais on essaye de l'approximer le mieux possible, avec de résultats qui ont une différence de moins de 1 % avec l'algorithme optimal.

La technique FirstIn-FirstOut est assez facile à implanter. Elle consiste à choisir comme victime, la page la plus anciennement chargée.

L'algorithme de remplacement de la page la moins récemment utilisée (*Least Recently Used*) est l'un des plus efficaces. Il nécessite des dispositifs matériels particuliers pour le mettre en œuvre. On doit notamment ajouter au tableau une colonne de compteurs. Par logiciel, on peut mettre en œuvre des versions dégradées.

5.7.3 Autres considérations

5.7.3.1 L'allocation locale ou globale

Lorsqu'on retire une page de la mémoire centrale, on peut choisir la plus ancienne :

- Du point de vue global (la plus ancienne du système);
- Du point de vue local (la plus ancienne du processus)D

En général, l'allocation globale produit de meilleurs résultats.

5.7.3.2 La prépagination

Lors du lancement d'un processus ou lors de sa reprise après une suspension, on provoque obligatoirement un certain nombre de défauts de pages. On peut essayer de les limiter en enregistrant, par exemple, l'ensemble de travail avant une suspension. On peut aussi essayer de le deviner. Par exemple, au lancement d'un programme, les premières pages de codes seront vraisemblablement exécutées.

6 Gestion de la mémoire secondaire : fichiers

Le S.E. doit pouvoir stocker à long terme de grandes quantités d'informations : en premier, ses propres modules, ensuite les programmes d'application, les bibliothèques de fonctions, les programmes et les données des utilisateurs.

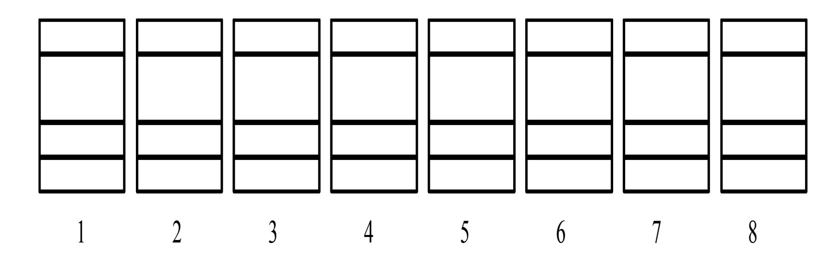
Les supports physiques : mémoire secondaire :

- Grande capacité,
- Faible coût.

Fonctions:

Organiser et gérer la mémoire secondaire en particulier faire la correspondance entre un fichier logique et un fichier physique.

Fichier: est une suite d'articles ou d'enregistrements spécifiques numérotés chacun avec des numéros différents et pouvant comporter chacun plusieurs champs.



Fichier physique : est la représentation physique de l'information.

Fichier logique : est un identificateur, qui désigne à la fois le fichier physique et la représentation logique, c.à.d. l'ensemble d'informations tel qu'il est vu par l'utilisateur.

6.1 Fichiers logiques

Les opérations de manipulation : créer, ouvrir, fermer, détruire, éditer, copier et renommer.

Les opérations d'accès : lire, écrire, insérer, détruire et retrouver.

Accès:

Le jeu d'instructions permettant d'accéder aux informations d'un fichier définit la structure logique dont les utilisateurs disposent pour organiser le fichier. Cette structure appelée le mode

d'accès aux fichiers varie d'un système à l'autre. Certains S.E. ne fournissent qu'une seule structure logique et donc qu'un seul mode d'accès.

6.1.1 Accès séquentiel

Les numéros d'ordre ne peuvent pas être utilisés par les fonctions d'accès. Ils permettent seulement de lire (ou écrire dans) l'article pointé, de pointer sur l'article suivant ou de pointer tout au début du fichier.

Exemple: lire lit le contenu de l'article courant et positionne le pointeur sur l'article suivant.

N.B. L'accès séquentiel est un mode hérité de l'époque où l'implantation physique des fichiers se faisait sur bande magnétique. Si la tête de L/E se trouve sur l'article 5, elle doit, pour lire l'article 10, se positionner successivement sur les articles 6, 7, 8, 9.

Utilisation: grand fichier et mises à jour peu fréquentes.

6.1.2 Accès indexé :

Permet d'accéder directement à un article, en utilisant certains champs qui sont appelés des clés.

Un cas particulier de l'accès indexé est l'accès direct où la clé d'un article est son numéro d'ordre. On le nommant, on accède directement à l'article.

Exemple : lire n, permet d'accéder directement à l'article correspondant.

6.2 Fichiers physiques

6.2.1 Organisation: implantation des articles

Elle dépend de la taille de l'information lue ou écrite en une seule opération par le périphérique d'E/S, le "bloc" (fixé par le matériel).

Dans UNIX les fichiers sont simplement des suites d'octets, dont chacun peut être adressé individuellement. L'enregistrement logique a la taille d'un octet. Les octets sont groupés par le système dans des blocs du disque (ex. 512 o/bloc).

La connaissance des tailles des articles et des blocs permet de connaître le nombre d'articles groupés dans un bloc.

L'organisation en zones de taille fixe induit naturellement le phénomène de fragmentation interne.

6.2.2 Gestion: méthodes d'allocation de mémoire secondaire

Le S.E. doit trouver un espace libre, cela va dépendre de l'allocation contiguë ou non.

6.2.2.1 Gestion des blocs libres

❖ Vecteur de bits (bit map)

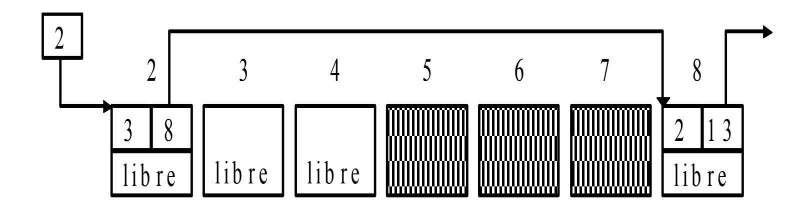
Vecteur contenant autant de bits que de blocs dans le disque. Chaque bit, associé à un bloc, est à 0 si ce bloc est libre, à 1 sinon.

❖ Liste chaînée :

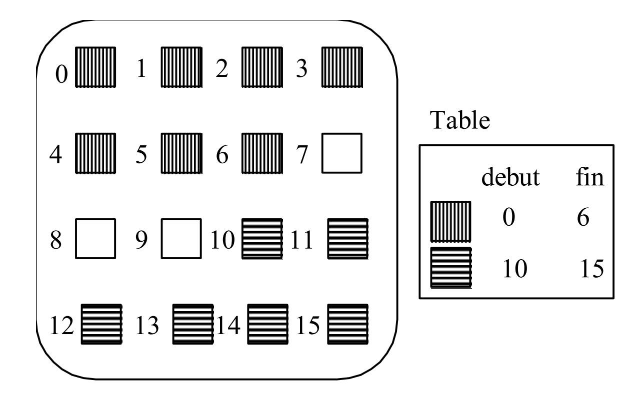
Chaque bloc libre contient le numéro du bloc suivant.

Inconvénient : Chercher n blocs nécessite le parcours de n blocs.

Amélioration : Le premier bloc de la zone libre contient le nombre de blocs libres et le numéro de la prochaine zone.



6.2.2.2 Allocation contiguë



On peut appliquer les méthodes de placement déjà présentées ainsi que les procédés de compactage.

Avantage :

- Permet d'accélérer les opérations de recherche,
- Implantation simple.

Inconvénients :

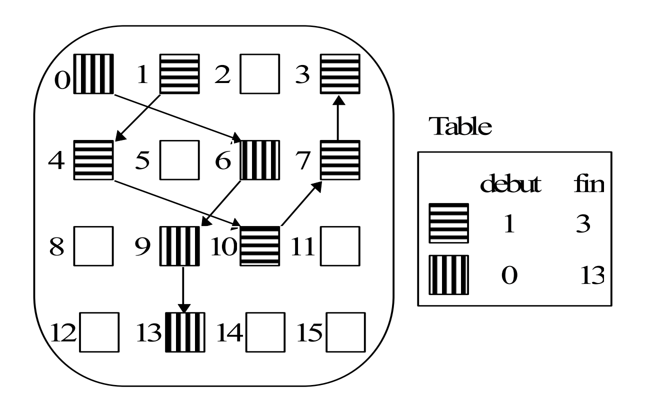
- Fragmentation interne et externe.
- A la création l'utilisateur doit spécifier la taille du fichier pour chercher un emplacement. Si pas d'espace assez grand, réorganisation du disque.
- On doit déplacer un fichier qui devient trop grand.

6.2.2.3 Allocation non contiguë

Deux méthodes : chaînée et indexée.

6.2.2.3.1 Allocation chaînée

Les blocs constituant un fichier donné sont chaînés, chaque bloc contenant le numéro du bloc suivant.



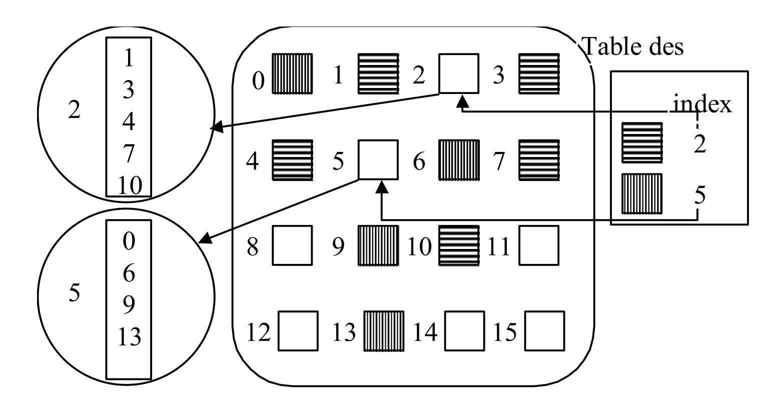
Avantages :

- Permet de faire évoluer la taille des fichiers,
- Ne pas avoir à déclarer la taille du fichier à sa création.
- Pas de compactage.

Inconvénients : Mode d'accès séquentiel et donc la mise à jour des pointeurs est assez coûteuse.

6.2.2.3.2 Allocation indexée

Table appelée bloc d'index est conservée dans un bloc spécial qui contient la liste des numéros de blocs utilisés par fichier.



Avantages :

- Accéder directement à un bloc quelconque du fichier,
- Augmentation dynamique de la taille du fichier,

Inconvénients : fragmentation interne provoquée par le stockage des tables d'index.

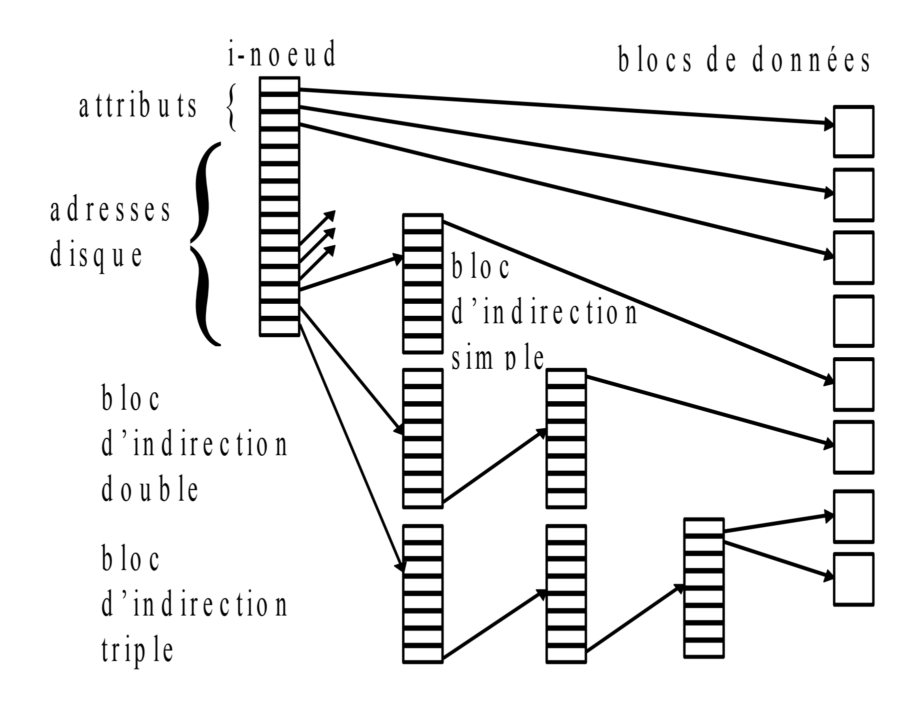
La solution : plusieurs niveaux d'index.

Un bloc qui contient 128 pointeurs permet d'indexer avec trois niveaux 128x128x128 = 2097152 blocs. L'inconvénient est qu'elle nécessite des accès supplémentaires.

UNIX utilise une quatrième solution mixte. Elle associe à chaque fichier une petite table appelée nœud d'information ou i-nœud. Cette table contient les attributs et les adresses sur le disque des blocs du fichier.

Un niveau d'index pour les petits fichiers et plusieurs niveaux pour les gros.

Taille fichier = $10+128+128^2+128^3$=2113674 blocs.

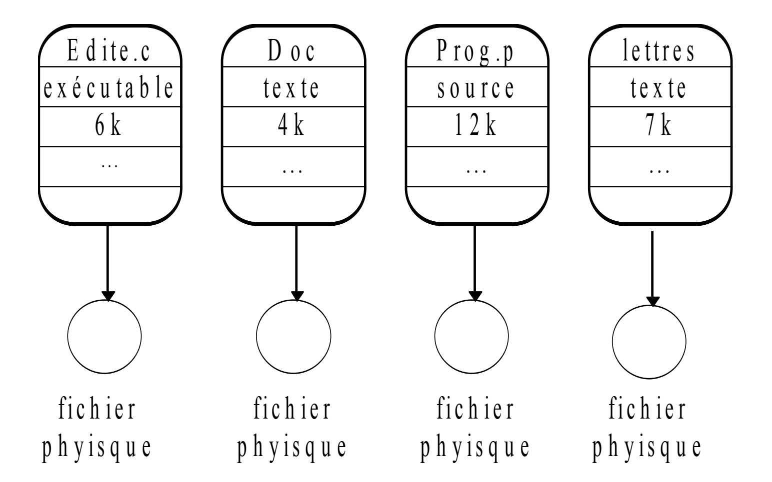


6.3 Correspondance fichiers logiques et physiques

6.3.1 Répertoires

Le S.E. doit assurer la correspondance entre le nom logique d'un fichier (l'identificateur connu de l'utilisateur) et son implantation physique. Il utilise pour cela une table appelée répertoire où, en regard du nom du fichier, il range un certain nombre d'informations :

- Nom logique,
- Le type,
- Adresse physique,
- La taille,
- Le caractère temporaire ou permanent,
- Des compteurs d'utilisations (nombre de lectures et d'écritures, de processus ayant ouvert le fichier, etc.).



a. Répertoire à un niveau

Un répertoire est donc une table qui établit une correspondance entre un nom et divers attributs.

Inconvénient : limitée

b. répertoire à deux niveaux

Chaque utilisateur a son propre répertoire avec un répertoire de niveau supérieur (master file directory), qui associe aux noms des utilisateurs les adresses de leurs répertoires propres.

Inconvénient : lorsqu'il s'agit de coopérer et de partager les fichiers.

Solution : les chemins (path). Le chemin de recherche (search path) permet de retrouver n'importe quel fichier.

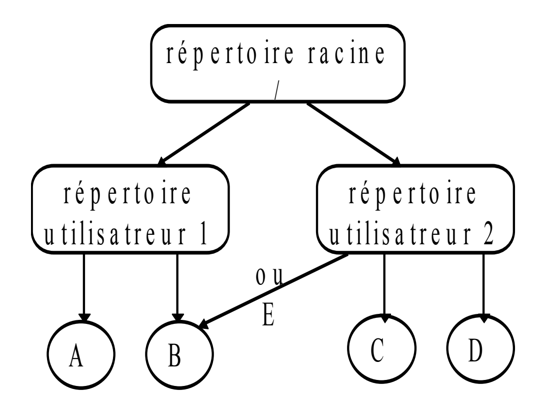
c. Répertoire à structure d'arbre

Extension du répertoire à deux niveaux.

d. Répertoires à structure de graphe sans cycle

Permet d'avoir le même fichier dans plusieurs répertoire sans avoir à le recopier en établissant un lien (link) entre l'originale et le répertoire dans le quel on veut créer le lien.

Un même fichier a plusieurs noms d'où problème à sa destruction et risque de création de cycle.



Pour l'implanter, on autorise qu'un répertoire puisse contenir comme entrée un pointeur vers un fichier (ou répertoire) qui n'est pas dans le sous-arbre dont il est la racine.

6.3.2 Opérations sur les fichiers

- Création
  - ✓ S'assurer que le nom n'existe pas (en explorant le répertoire),
  - ✓ Ajouter une entrée si aucune n'est libre,
  - ✓ Réserver la mémoire secondaire suffisante,
  - ✓ Enregistrer les informations dans les champs correspondants au fichier dans le répertoire.
- Destruction
  - ✓ Chercher si le nom existe dans le répertoire sinon erreur,
  - ✓ Détruire l'espace physique alloué,
  - ✓ L'enlever du répertoire.
- Ouverture
  - ✓ Chercher l'identificateur du fichier dans le répertoire,
  - ✓ Vérifier les autorisations d'accès en L/E.
  - ✓ Si le fichier peut être ouvert, le S.E. doit, pour ne pas avoir à le refaire à chaque accès au fichier :

    Déterminer l'adresse physique en mémoire secondaire où il est rangé,

    Construire un **bloc de contrôle de fichier** (**BCF**) qui sera garder en M.C. et qui contient :
    - ⇒ Identificateur,
    - ⇒ Adresse physique du descripteur du périphérique d'E/S réalisant les transferts entre la M.C. et la M.S.,
    - ⇒ Adresse du bloc du fichier, et
    - ⇒ Dans le cas d'accès séquentiel l'adresse du bloc pointé.
  - ✓ Lorsque plusieurs utilisateurs ouvrent en même temps un fichier, le S.E. attribue à chacun d'entre eux un BCF et fournit à chacun d'eux un pointeur appelé **pointeur de connexion**. Ceci permet des accès indépendants et d'accélérer les opérations de manipulation.
- Fermeture

  On détruit simplement le BCF

- Accès aux articles

Les opérations de L/E/M consistent à chercher les le bloc de la M.S. qui contient l'article, puis à effectuer l'opération qui entraîne le transfert entre la M.S. et la M.C.

L'article est repéré par son numéro d'ordre pour l'accès direct (adresse relative par rapport au début du fichier) ou est le suivant de l'article courant pour l'accès séquentiel.

6.4 Protection

- Protection contre accident
  - ✓ Sauvegarde périodique
  - ✓ Sauvegarde incrémentale.
- Protection des accès
  - ✓ mots de passe
    - ❖ mot de passe simple,

- ❖ mot de passe contenant un nombre choisi au hasard et le tout codé,
- ❖ mot de passe changé régulièrement par le S.E.,
- ❖ mot de passe avec questions personnelles stockées à l'avance et de manière codée,
- ❖ choix d'un algorithme par l'utilisateur(x^2),
- ❖ identification physique.
  - carte magnétique avec mot de passe,
  - empreintes digitales,
  - intonations vocales,
  - signature (vérifier le mouvement et non la signature),
  - longueur des doigts.
- ❖ identificateur de l'utilisateur.

- Protection contre attaque
  - ✓ Ver d'Internet

La plus grande faille informatique de tous les temps, s'est produite le 2 novembre 1988 lorsqu'un étudiant de l'université de Cornell, Robert Tappan Morris, a introduit un ver dans le réseau Internet qui connecte des milliers de machine à travers le monde et qui a causé leur effondrement. Le programme ver qu'il a écrit, qui exploitait les erreurs de Unix se dupliquait en quelque seconde sur les machines auxquelles il accédait. Il était composé en un programme en C de 99 lignes qui était compilée et exécutée sur le système attaqué. En cours d'exécution il se connectait à la machine d'où il provenait, transférait le ver et l'exécutait. Après avoir tenter de cacher son existence le ver examinait les tables de routage de la machine hôte pour chercher de nouvelle victime.

  - ✓ Virus

Un virus est un morceau de programme rattaché à un programme normal qui tente de s'introduire dans d'autres programmes. Le créateur du virus, crée un programme utile(jeu) qui dissimule le code du virus. Lorsqu'on exécute le jeu, il commence par examiner les programmes installés sur le disque dur pour déterminer s'ils sont déjà infecter. Lorsqu'un programme non infecter est trouvé, le code du virus est rajouté à la fin du programme et première instruction est remplacée par un branchement vers le code du virus. Lorsque le code du virus termine son exécution, il exécute l'instruction qui était la première du programme et effectue un branchement sur la deuxième instruction du programme. De cette manière, à chaque exécution du programme, le virus tente d'infecter de nouveaux programmes.

6.5 Les Entrées Sorties

6.5.1 Hardware

6.5.1.1 Les périphériques d'E/S

- périphériques blocs

Ils mémorisent les informations dans des blocs de taille fixe, chaque bloc ayant une adresse fixe et de taille variant entre 128 et 1024.

Exemple : les disques sont des périphériques blocs “bandes”.

- périphériques caractères

  Ils acceptent un flot de caractères sans se soucier d’une quelconque structure de bloc.

Exemple: les imprimantes, les terminaux, les interfaces réseau les souris, etc.

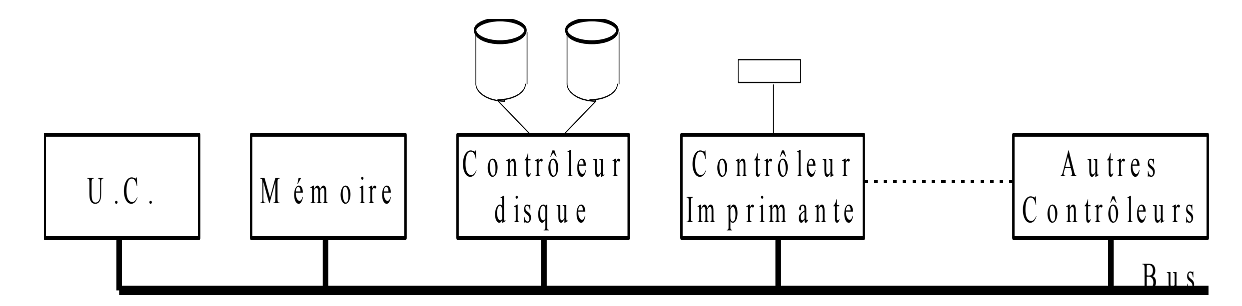
Le système de fichier manipule des périphériques blocs abstraits et laisse les pilotes de périphériques (devices drivers), qui est un logiciel de plus bas niveau, se charger de tout ce qui dépend des périphériques.

6.5.1.2 Les contrôleurs de périphériques

C’est le hardware contrôlant le périphérique (carte).

Un contrôleur peut accepter 2, 4 ou 8 périphériques identiques.

N.B. Le S.E. communique avec les contrôleurs et non avec les périphériques au moyen d’un bus unique.



L’interface entre le contrôleur et le périphérique est de bas niveau.

- **Disque** : Un disque peut, par exemple, être formaté en pistes de 8 secteurs de 512 o.

Le disque fournit une série de bits constitués d’un préambule, des 4096 bits d’une piste et d’un code correcteur d’erreur (error-correcting code, ECC).

Le préambule est écrit lors du formatage du disque et contient les numéros de cylindre et de secteur, la taille des secteurs et d’autres données de ce type.

Le travail du contrôleur est de regrouper ce flot de bits en série dans un bloc d’octets en corrigeant les erreurs si nécessaire. Le bloc est constitué bit par bit dans un tampon du contrôleur.

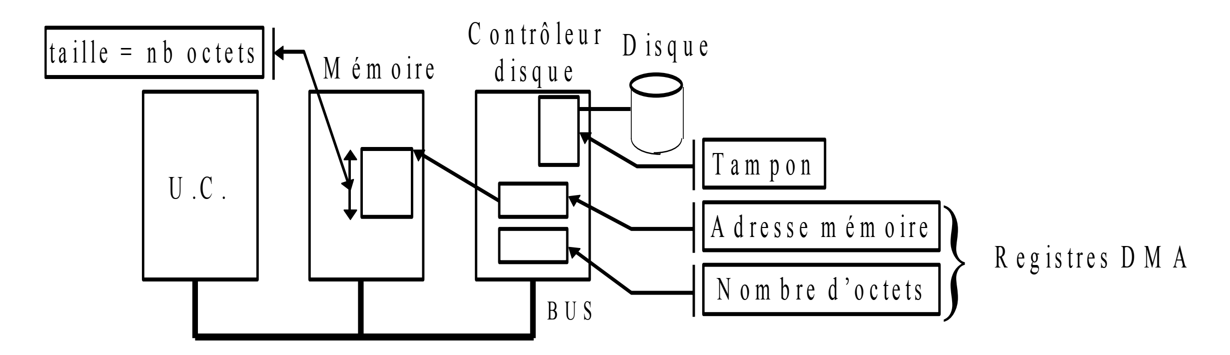
Si aucune erreur n’est détectée, il est recopié dans la M.C.

- **Terminal** : C'est un contrôleur série de bas niveau. Il lit dans la mémoire les octets qui contiennent les caractères à afficher et génère des signaux pour moduler le faisceau d'électrons.

Le contrôleur communique avec le processeur par l'intermédiaire des registres. Le S.E. effectue les E/S en :

- écrivant des commandes dans les registres,
- Le contrôleur accepte ou refuse une commande (lorsqu'elle est accepter, le processeur peut effectuer un autre travail),
- Lorsque la commande est exécutée, le contrôleur envoie une interruption pour permettre au S.E. de réquisitionner le processeur afin de tester les résultats de l'opération.

L'accès direct à la mémoire (DMA)



Le contrôleur lit le bloc du périphérique, place les données dans le tampon et vérifie le total de contrôle (checksum), ensuite :

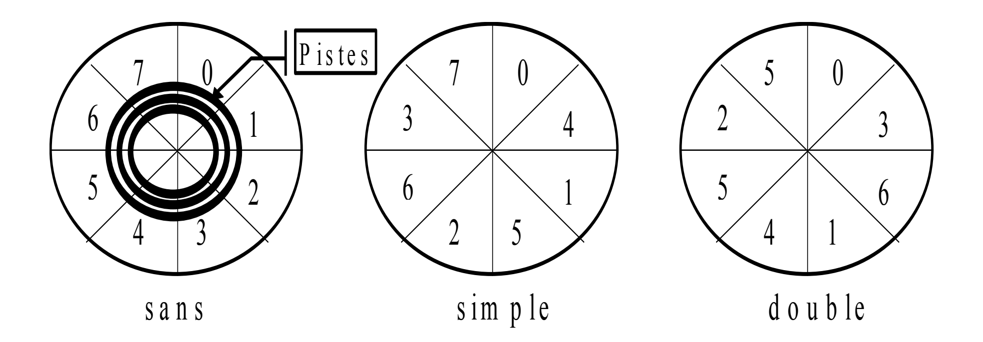
- copie le premier octet dans la M.C. à l'adresse spécifiée au DMA,
- incrémente cette adresse,
- décrémente le compteur du compteur du DMA du nombre d'octets transférés,
- répète cette opération jusqu'à ce que le compteur du DMA atteigne la valeur zéro.

Pourquoi le transfère se fait d'abord vers le tampon ensuite vers la mémoire ? (bus commun)

Lecture du disque :

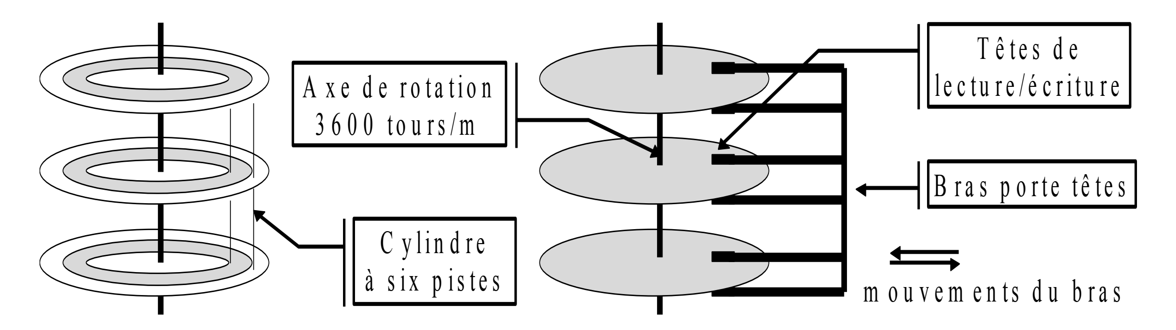
Pendant le transfère des données du contrôleur vers la M.C., le disque continuant de tourner, le secteur suivant et lu et ses données sont transférées au contrôleur qui est occupé à transférer les données du tampon vers la M.C., elles sont, par conséquent, perdues. Et comme les données ont des chances d'être sur la même piste (2 secteurs consécutifs) il faut donc 2 rotations pour lire une piste. Si le temps de transfère d'un bloc du contrôleur vers la M.C. est supérieur au temps de lecture d'un secteur du disque, il faudra peut-être lire un bloc et sauter les deux suivants ou plus.

Solution entrelacement (interleaving)



Si un contrôleur lit un bloc/2, il faut 8 rotations pour lire 8 blocs consécutifs.

Un bloc = 1 ou plusieurs secteurs.



Temps d'accès, plusieurs étapes :

- attente dans la file d'attente du dispositif d'E/S si celui-ci n'est pas disponible,
- déplacement du bras pour le positionner sur le bon cylindre. Ce temps est connu comme le **temps de recherche**,
- amener le secteur concerné sous la tête de L/E avec une durée appelée **temps de latence**, enfin
- le transfert depuis ou vers la M.C. est effectué avec un délai appelé **temps de transfert**.

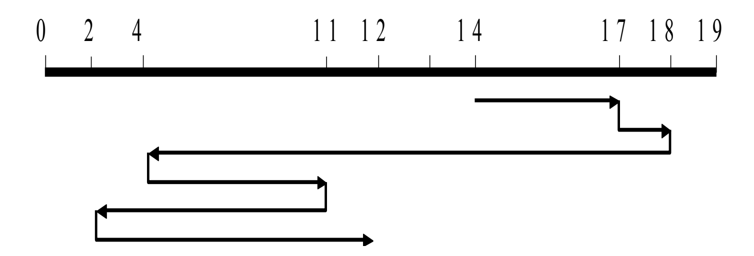
Le temps d'accès proprement dit c'est à dire regroupant toutes ces opérations ne tient pas compte des temps d'attente dans les files.

6.5.2 Algorithmes d'ordonnancement

6.5.2.1 Le premier arrivé (FIFO)

Le plus simple mais pas le plus efficace.

Exemple : On veut lire les pistes 17, 18, 4, 11, 2 et 12 sur un disque de 20 pistes et dont la tête se trouve sur la piste 14.



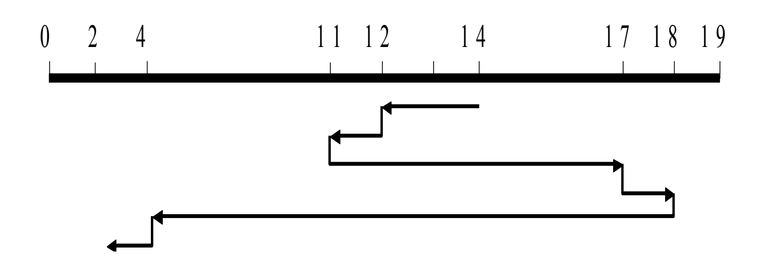
Le déplacement : d = (17-14)+(18-17)+(18-4)+(11-4)+(11-2)+(12-2)=44 pistes.

Pour minimiser cette valeur on peut traiter 4 et 2 ensemble avant 11 et 12 ce qui donnerait

d = 30.

6.5.2.2 Le plus court temps de recherche (PCTR)

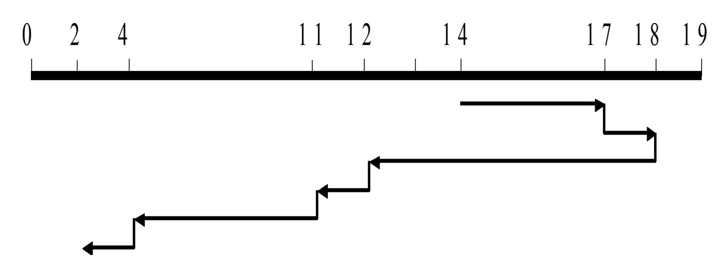
Permet de regrouper les pistes proches ce qui implique la prochaine requête à traitée est celle pour laquelle le déplacement de la tête est minimal.



d = (14-12)+(12-11)+(17-11)+(18-17)+(18-4)+(4-2)=26 pistes.

Inconvénient :

- Risque de famine, puisque si on reçoit des requêtes dont la distance est toujours la plus petite, on fait attendre indéfiniment les autres.
- Cette méthode n'est pas optimale non plus en effet :



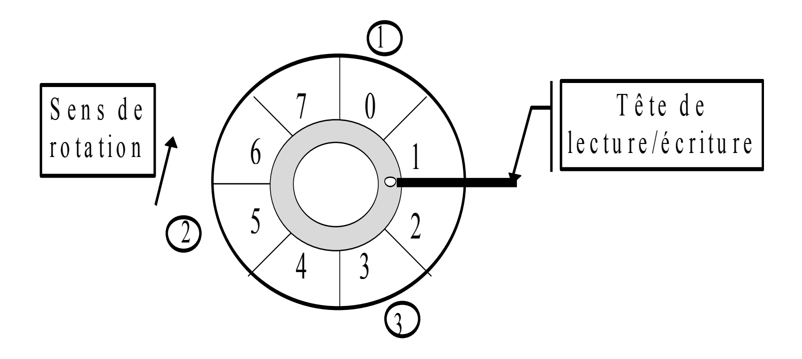
d = (17-14)+(18-17)+(18-12)+(12-11)+(11-4)+(4-2)=20 pistes.

6.5.2.3 Balayage (SCAN)

Pour remédier à l'inconvénient précédent on utilise l'algorithme SCAN qui parcourt toutes les pistes dans un sens, répond aux requêtes des pistes se trouvant sur son passage et repart dans l'autre sens, une fois arrivé au bout.

a. L'algorithme lOOK fonctionne de la même manière, sauf qu'il ne va pas jusqu'au bout de son parcourt s'il n'y a plus de requête en attente.
b. L'algorithme C-LOOK et C-SCAN (circulaire) traitent les pistes d'une manière circulaire, c.à.d qu'ils considèrent que la dernière piste est adjacente à la première. Dés qu'ils arrivent à la dernière ils repartent tout de suite à la première sans traiter de requêtes.
c. Plus court temps de latence (PCTL). Il sélectionne les requêtes concernant le secteur le plus proche de la position courante de la tête, en tenant compte du sens de rotation.

Donc : À chaque secteur d'un même cylindre est associée une file d'attente des requêtes pour ce secteur.



6.5.3 Les pilotes de périphériques (device driver)

Un pilote traite un type de périphérique. Un contrôleur possède un ou plusieurs registres de commande. Les pilotes envoient ces commandes et vérifient leur bon acheminement.

Exemple : pilote du disque : c'est la partie software du système qui connaît les registres du contrôleur du disque, les secteurs, les pistes, les cylindres, les têtes, le déplacement du bras, le facteur d'entrelacement, les moteurs, le temps de positionnement des têtes et tous les autres mécanismes qui permettent le fonctionnement du disque.

Un pilote de disque doit :

- déterminer où se trouve le bloc à chercher,
- vérifier si le moteur tourne,
- vérifier si le bras est positionné si non le faire,
- remplir les registres du contrôleur pour exécuter les requêtes,
- se bloquer pour attendre la fin de l'opération,
- continuer après la réception d'une interruption qui indique la fin
- vérifier s'il y a erreur,
- si pas d'erreur, transmettre le bloc lu au demandeur,
- renvoyer un code d'erreur.
- exécuter la prochaine requête ou se bloquer en attendant.